

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Направление: 09.03.03 – Прикладная информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
**РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ
СИСТЕМЫ ДЛЯ АНАЛИЗА И МОНИТОРИНГА ЦЕН
ИНТЕРНЕТ-МАГАЗИНОВ**

Работа завершена:

Студент 4 курса

Группы 09-751

«4» июня 2021 г.

Р. Зеленов

Зеленов Р.А.

Работа допущена к защите:

Научный руководитель

к.ф.-м.н., доцент кафедры

технологий программирования

«4» июня 2021 г.

И.С. Балафендиева

Балафендиева И.С.

Заведующий кафедрой

к.э.н., доцент

«4» июня 2021 г.

Г.З. Вахитов

Вахитов Г.З.

Содержание

Аннотация	3
Введение.....	5
1. Исследование предметной области анализа и мониторинга цен интернет-магазинов и описание средств разработки	8
1.1. Анализ предметной области	8
1.2. Обзор сервиса «Яндекс.Маркет»	9
1.3. Обзор ресурса «Pepper.ru».....	13
1.4. Описание программных средств реализации	14
2. Проектирование многопользовательского веб-приложения	16
2.1. Составление технического задания.....	16
2.2. Проектирование базы данных и пользовательского интерфейса	18
3. Разработка многопользовательского веб-приложения	21
3.1. Реализация серверной части веб-приложения	21
3.2. Реализация пользовательской части информационной системы.....	31
3.3. Разработка модулей для автоматизированного мониторинга цен	45
4. Тестирование многопользовательского веб-приложения.....	49
4.1. Тестирование пользовательского интерфейса веб-приложения	49
4.2. Тестирование функциональной части веб-приложения.....	51
Заключение	53
Список использованных источников	57
Приложения	

Аннотация

Выпускная квалификационная работа по теме «Разработка автоматизированной информационной системы для анализа и мониторинга цен интернет-магазинов» состоит из 62-ух страниц текста, 33-ех рисунков, 9-ти листингов кода и 22-ух использованных источников. Структура работы включает введение, 4 главы, заключение, список использованных источников и приложения.

Целью работы является создание автоматизированной информационной системы для анализа и мониторинга цен интернет-магазинов.

Первая глава содержит анализ предметной области, обзор аналогов и описание программных средств реализации. Вторая глава включает в себя составление технического задания, а также проектирование базы данных и пользовательского интерфейса. В третьей главе описаны процессы разработки серверной части веб-приложения, пользовательской части информационной системы и модулей для автоматизированного мониторинга цен интернет-магазинов. Четвертая глава посвящена тестированию разработанной автоматизированной информационной системы, в ней описаны основные принципы тестирования. В заключении перечислены основные результаты выполнения выпускной квалификационной работы.

Ключевые слова: веб-приложение, автоматизированная информационная система, мониторинг, цен, интернет-магазин, Java, Python, PostgreSQL, Vue.js

Abstract

The final qualification work on the topic «Development of an automated information system for the analysis and monitoring of prices for online stores» consists of 62 pages of text, 33 drawings, 9 code listings and 22 used sources. The structure of the work includes an introduction, 4 chapters, a conclusion, a list of references and applications.

The aim of the work is to create an automated information system for the analysis and monitoring of prices for online stores.

The first chapter contains an analysis of the subject area, an overview of analogs and a description of software implementation tools. The second chapter includes the preparation of the terms of reference, as well as the design of the database and user interface. The third chapter describes the processes of developing the server part of a web application, the user part of the information system and modules for automated monitoring of prices for online stores. The fourth chapter is devoted to testing the developed automated information system; it describes the main principles of testing. In the conclusion, the main results of the final qualifying work are listed.

Keywords: web application, automated information system, monitoring, prices, online store, Java, Python, PostgreSQL, Vue.js

Введение

С каждым днем появляется все больше интернет-магазинов, именно благодаря их активному развитию сегодня почти каждый так или иначе взаимодействовал с их сайтами в интернете, заказывая товары прямо из дома. Это действительно намного удобнее, чем выбирать вещь в самом магазине и выгодно всем. Перед покупателем открывается огромный выбор интернет-магазинов и товаров, магазины, в свою очередь, не нуждаются в постоянном расширении торговых площадей и расположении их в различных частях города. К тому же отпадает необходимость размещать все товары на небольшой площади физического магазина, ведь можно разместить тысячи или даже десятки тысяч товаров на одном интернет-ресурсе. Уже сегодня каждый крупный и средний ритейлер имеет в своем распоряжении быстрый, удобный и интуитивно-понятный интернет-магазин, где каждый желающий может заказать понравившийся товар буквально в несколько кликов. По статистике Росстата доля интернет-торговли в России удвоилась за 2020-ый год. Однако такое разнообразие товаров и интернет-магазинов привело к тому, что для поиска наилучшего предложения может потребоваться не один час. Регулярное снижение и повышение цен в интернет-магазинах сбивают людей с толку. Постоянные распродажи товаров с предварительным увеличением цены на размер будущей скидки вводят людей в заблуждение, позволяя зарабатывать интернет-магазинам, но не экономить обычным пользователям. А при реальном снижении цены на интересующий пользователя товар, он сможет узнать об этом далеко не сразу и, вероятно, пропустит выгодное для него предложение. Невозможно целыми днями просматривать десятки страниц в надежде успеть приобрести товар по выгодной цене. Поэтому большинство людей принимают решение о покупке той или иной вещи интуитивно.

Естественно, такой подход редко приносит желаемый результат. Многие пользователи с удивлением обнаруживали купленный недавно товар по более выгодной цене в том же магазине. А также регулярно сталкивались с каким-то выгодным товаром по низкой цене, которого уже нет в наличии. Более того, часто пользователям интернет-магазинов просто хочется приобрести что-то выгодное и совсем необязательно, что они в этом предмете нуждаются прямо сейчас. Именно из-за этого появились информационные системы, собирающие и агрегирующие в себе данные с десятков интернет-магазинов, позволяя пользователям ресурса узнавать о скидках на различные товары. Однако такие информационные системы не позволяют просматривать все товары, представленные в определенном магазине, множество из них функционирует в ручном режиме, из-за чего данные в таких информационных системах имеют очень низкую актуальность. А также в них не представлены крупные российские и международные ритейлеры. Что еще более важно, подавляющее большинство таких информационных ресурсов вообще не имеет никаких механизмов оповещения пользователя о появлении выгодного предложения с интересующим его товаром. Хоть такие ресурсы и упрощают поиск выгодных предложений, но они, в большинстве, не предоставляют или не собирают статистику о динамике цены на товар, что не позволяет пользователю совершить оптимальный выбор.

Актуальность данной работы заключается в том, что на сегодняшний день не существует автоматизированной информационной системы, которая бы собирала данные о ценах как с малых интернет-магазинов, так и с крупных российских и международных ритейлеров, а также предоставляла возможность отслеживания интересующего пользователя товара и получения уведомления об изменении цены в мессенджер Telegram. Полная автоматизация процесса получения цены каждого товара позволит обеспечить высокую актуальность данных и своевременно оповестить

пользователя о выгодном для него предложении. В процессе консультации с научным руководителем была поставлена цель и выделены задачи. Была налажена регулярная коммуникация с научным руководителем для уточнения вопросов и консультаций. Из-за текущей эпидемиологической ситуации все консультации проводились в удаленном формате.

Целью ВКР является создание автоматизированной информационной системы для анализа и мониторинга цен интернет-магазинов.

Задачи выпускной квалификационной работы:

1. Провести анализ предметной области;
2. Провести обзор существующих аналогов;
3. Составить техническое задание к функциональной части и интерфейсу;
4. Описать программные средства реализации информационной системы;
5. Спроектировать базу данных и пользовательский интерфейс;
6. Разработать серверную часть;
7. Протестировать информационную систему и отладить ее.

Выпускная квалификационная работа состоит из 4-ех глав, введения, заключения, списка использованной источников и приложений. В работе описан весь процесс проектирования, конструирования и разработки системы, представлены скриншоты страниц веб-приложения, описания классов, репозиториев и контроллеров, а также вставлены наиболее важные части кода приложения.

1. Исследование предметной области анализа и мониторинга цен интернет-магазинов и описание средств разработки

1.1. Анализ предметной области

При анализе интернет-магазинов были выделены закономерности об их устройстве. В каждом интернет-магазине определены глобальные категории товаров верхнего уровня. Каждая такая категория является многоуровневым иерархическим меню с подкатегориями. При выборе любой из категорий отображаются только те товары, которые принадлежат выбранной категории. Все интернет-магазины предоставляют пользователям одинаковый объем информации о товаре, а именно: данные о характеристиках товара, его изображение, цену на текущий момент в интернет-магазине, отзывы и рейтинг товара, как правило по пятибалльной шкале.

В подавляющем большинстве интернет-магазинов пользователи могут оценить товар и оставить комментарий о нем. Таким образом происходит сбор информации о качестве представленного товара, а также формируется его рейтинг. Это также приводит к большей заинтересованности товаром других пользователей, так как человек с большей вероятностью купит товар с хорошими отзывами и высоким показателем рейтинга, чем без отзывов вовсе, пытаясь минимизировать риск неверного решения. В качестве дополнительного стимулирования спроса на определенные позиции интернет-магазины могут снижать цены на эти товары.

Основной сценарий пользователя в интернет-магазине заключается в том, чтобы найти интересующий его товар с наибольшим количеством положительных отзывов и высоким рейтингом за наименьшую стоимость. Однако пользователь не может быть полностью уверен в своем выборе, хотя бы потому, что цена на выбранный товар может меняться по несколько раз за день, но ни один интернет-магазин не предоставляет график динамики цены товара.

1.2. Обзор сервиса «Яндекс.Маркет»

На сегодняшний день существует несколько информационных систем, которые осуществляют мониторинг цен различных интернет-магазинов. Наиболее известным представителем является информационная система «Яндекс.Маркет». Этот интернет-ресурс предоставляет пользователям информацию о текущих ценах на различные товары, собирая и агрегируя данные с нескольких десятков интернет-магазинов. «Яндекс.Маркет» имеет внушительный список категорий, так как агрегирует данные с интернет-магазинов разной направленности, что ощутимо усложняет поиск нужной категории пользователями. Навигация в списке категорий не интуитивна и слишком усложнена. К примеру, в разделе «Электроника» представлено более 130 подкатегорий, названия которых расположены не в лексикографическом порядке, что дополнительно усложняет поиск.

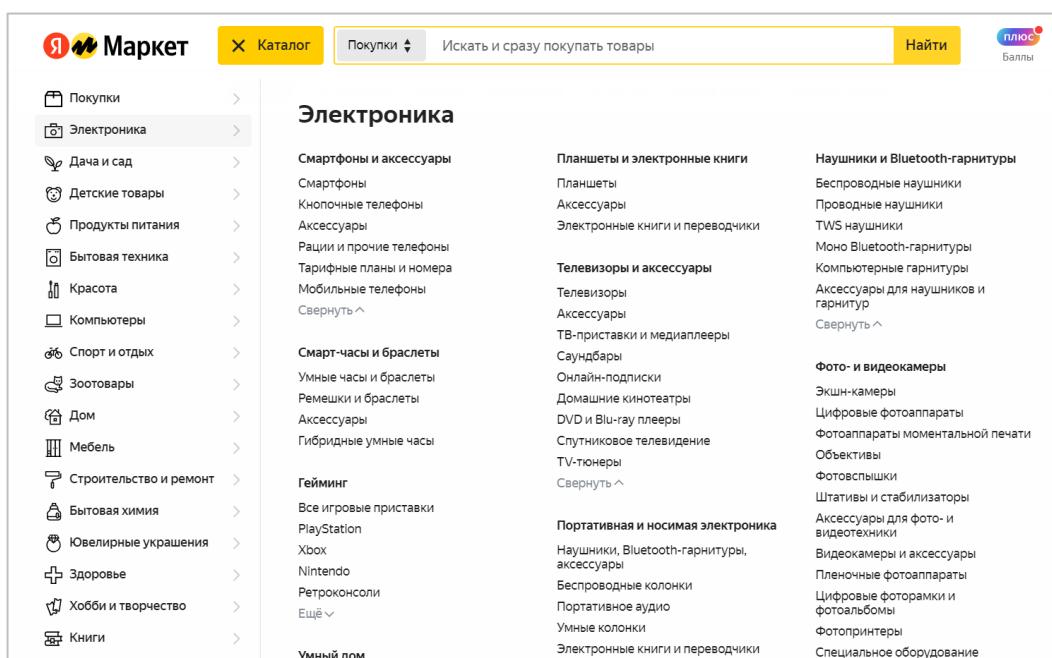


Рисунок 1.1. Навигация в категории «Электроника»

Информационная система имеет поисковую строку для быстрого нахождения товара по ключевым словам. Однако поиск часто встречающихся понятий без четкого указания модели или названия товара возвращает

несколько тысяч результатов, что является избыточным для подавляющего большинства пользователей.

The screenshot shows a search interface for 'Ноутбуки' (Laptops). At the top, there are navigation links: 'Казань' (Kazan), 'Покупки' (Purchases) which is highlighted in blue, 'Супермаркет' (Supermarket), 'Экспресс' (Express), and 'Электроника' (Electronics). Below this, a breadcrumb trail shows 'Покупки · Компьютеры · Ноутбуки и планшеты · Ноутбуки'. There is also a link 'Искать везде' (Search everywhere). The main heading 'Найдено 2186 товаров' (Found 2186 items) is displayed prominently. The background is white with some light gray horizontal lines.

Рисунок 1.2. Поиск по ключевому слову «Ноутбук»

Системой предусмотрены различные виды фильтрации товаров, для уточнения поискового запроса, но даже их представлено несколько десятков. Объединение огромного количества позиций из каждого магазина из схожих категорий привело к сильной перегрузке интерфейса.

The screenshot displays a list of filters for a specific product. It includes sections for 'Цена, ₽' (Price), 'Включено в стоимость' (Included in price), 'Скидки и акции' (Discounts and promotions), 'Производитель' (Manufacturer), 'Максимальная скорость электрического велосипеда, км/ч' (Maximum speed of an electric bicycle, km/h), 'Мощность двигателя, Вт' (Motor power, W), 'Время зарядки, ч' (Charging time, hours), 'Тип аккумулятора' (Battery type), 'Уровень каретки' (Cart level), 'Конструкция каретки' (Cart construction), and 'Тип посадочной части вала каретки' (Type of bearing seat). Each filter section contains input fields for minimum and maximum values or checkboxes for specific options. The background is white with light gray horizontal lines.

Рисунок 1.3. Список фильтров для уточнения запроса

Несмотря на это, «Яндекс.Маркет» позволяет удобно просматривать выбранный товар. Прямо на странице товара указаны характеристики,

пользовательские оценки и отзывы. Не менее важным элементом является график цены, который позволяет наглядно увидеть максимальную и минимальную цены на этот товар за определенный промежуток времени и сделать вывод о динамике цены.



Рисунок 1.4. График динамики цены

Важным преимуществом является возможность сравнить цены в различных интернет-магазинах за один и тот же товар. Также предоставляется информация о стоимости и средней продолжительности доставки, способе оплаты и рейтинге магазина.

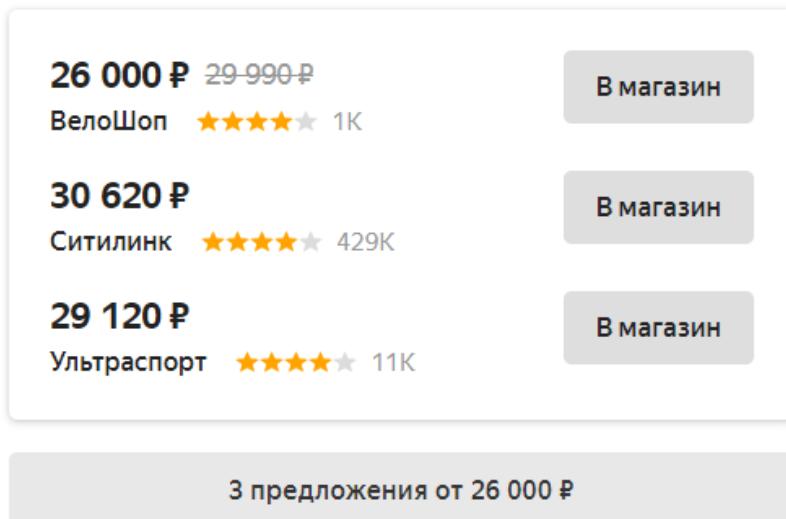


Рисунок 1.5. Список цен за аналогичный товар в различных магазинах

Еще одним преимуществом является возможность задать вопрос о товаре, на который могут ответить пользователи, купившие этот товар ранее, либо сам производитель.

Анастасия Б. 7 месяцев назад

Из какого материала выполнены боковые стенки холодильника?

[Ответить](#) 0

Екатерина Шутова 3 6 месяцев назад

Тонкий метал

Рисунок 1.6. Пример вопроса от пользователя и ответа

Цены товаров, представленных на площадке, обновляются автоматически, что позволяет постоянно поддерживать их в актуальном состоянии. Однако совокупность представленных в системе интернет-магазинов не включает популярных крупных международных и российских ритейлеров. Поэтому зачастую выбор пользователя, сделанный на основании информации из системы «Яндекс.Маркет», не является оптимальным. В информационной системе отсутствует возможность получения уведомлений о снижении цены в популярные мессенджеры. В этом случае пользователь сервиса может пропустить товар по выгодной для него цене. Более того, еще одним существенным недостатком является приоритетное продвижение товаров, которые можно купить именно через «Яндекс.Маркет». Первая страница выдачи, как правило, состоит из тех вещей, которые выгодны самому сервису. Таким образом посетитель информационного ресурса приобретает товар, который выгоден сайту «Яндекс.Маркет», а не самому пользователю.

1.3. Обзор ресурса «Pepper.ru»

Достаточно известной является информационная система «Pepper.ru». На этом веб-ресурсе регулярно публикуются товары со скидкой из множества интернет-магазинов. Помимо этого, пользователи могут оценить скидку. На основании их голосов и комментариев формируется рейтинг предложения, что позволяет пользователям дополнительно оценить, насколько выгодным является предложение с точки зрения других людей. Система фильтрации позволяет легко и быстро найти определенный товар или категорию товаров.



Рисунок 1.7. Карточка товара с высокой оценкой от пользователей

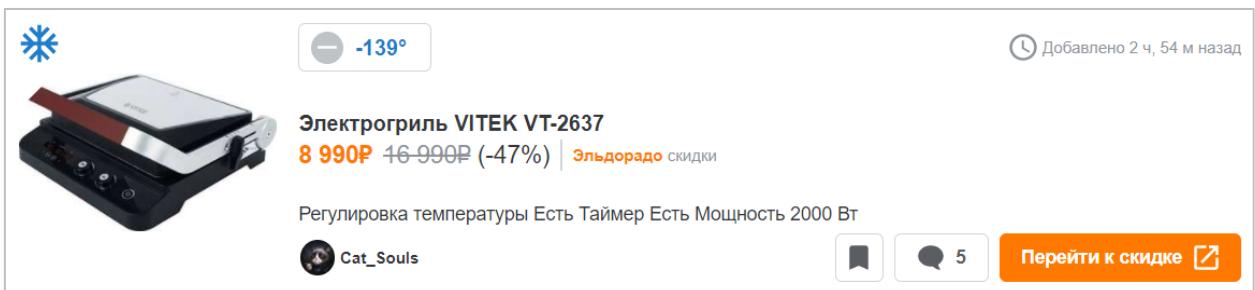


Рисунок 1.8. Карточка товара с низкой оценкой от пользователей

Главным недостатком является то, что скидки вручную предлагаются пользователями, либо публикуются администрацией сайта. Из-за этого пользователи регулярно сталкиваются с низкой актуальностью данных, так как невозможно поддерживать предложение в актуальном состоянии на протяжении долгого времени вручную, описание товара часто

является ошибочным, неточным или вовсе отсутствует. Невозможно оценить динамику цены товара. Помимо этого, в системе не предусмотрено возможности оповещения пользователя о скидках в интересующих его категориях и о снижении цены отслеживаемого товара.

1.4. Описание программных средств реализации

Для разработки REST API был выбран язык программирования Java и фреймворк Spring. Исходя из чего была выбрана среда разработки IntelliJ IDEA. IntelliJ IDEA – высокотехнологичная среда разработки, позволяющая удобно и просто разрабатывать приложения и системы. Данная среда поддерживает множество фреймворков и технологий, что значительно упрощает работу в ней и позволяет эффективно работать с кодом. Spring – удобный фреймворк с рядом предопределенных расширений и технологий, позволяющий оптимизировать создание веб-приложений [13]. Для создания веб-интерфейса приложения был выбран фреймворк Vue.js, который легко интегрируется в проекты, а также позволяет разделить логику, стили и разметку страницы. Это упрощает проектирование интерфейса. Вместе с тем, он позволяет импортировать готовые компоненты в свой проект, что упрощает работу над приложением. Также к Vue.js были подключены графические фреймворки Vuetify и Bootstrap, которые позволяют воспользоваться готовыми компонентами для конструирования лаконичного и стильного интерфейса [14].

Для парсинга страниц интернет-магазинов был выбран язык программирования Python 3. Для него создано огромное количество сторонних библиотек, позволяющих взаимодействовать с веб-страницами, а также парсить их. Взаимодействие реализовано через формирование POST и GET запросов к REST API серверу. В качестве системы управления базами данных была выбрана PostgreSQL. Она обладает надежными и быстрыми механизмами обработки транзакций, а также поддерживает базу данных

неограниченного размера, что является очень важным при создании систем мониторинга каких-либо данных.

2. Проектирование многопользовательского веб-приложения

2.1. Составление технического задания

В результате анализа предметной области, цели и поставленных задач возникла необходимость разработки многопользовательского веб-приложения для автоматического анализа и мониторинга цен интернет-магазинов. Часть системы, отвечающая за REST API, должна быть написана на языке программирования Java, а часть, отвечающая за парсинг интернет-магазинов на Python. Взаимодействие Python с базой данных должно быть реализовано путем формирования и отправки POST и GET запросов.

Система должна представлять собой веб-приложение, с удобной навигацией по магазинам, возможностью просмотра товаров, их цен. Также должна быть реализована возможность подписки пользователя на интересующий товар с оповещением об изменении цены на товар в мессенджер Telegram. Мониторинг и анализ цен в интернет-магазинах должен быть полностью автоматизирован, данные должны поддерживаться в актуальном состоянии. Система должна автоматически обновлять цены по каждому товару, который присутствует в базе данных, не реже 1-ого раза в час, тем самым поддерживая информацию в актуальном состоянии. В процессе парсинга система должна определять, продается ли товар со скидкой в текущий момент и парсить цену без скидки и с ней автоматически. Информацию о всех ценах система должна помещать в базу данных. Система должна хранить информацию о предыдущих ценах на данный товар для последующего анализа. Также цена должна быть привязана к дате для упорядочивания стоимости товара и составления динамики цены на конкретный товар. Объем получаемых из интернет-магазина данных должен включать информацию о самом магазине, список всех категорий в нем, список товаров в каждой категории, информацию о каждом товаре, список характеристик, а также цену каждого товара. Системой предусмотрено 2

уровня привилегий: пользователь и администратор. Администратор должен иметь доступ в admin-панель, в которой предусмотрена возможность добавления, удаление и редактирования информации интернет-магазина. Необходимо предусмотреть отказоустойчивость информационной системы в случае передачи некорректных данных. Количество ошибок при парсинге страниц товаров модулями не должно превышать 0,5% от общего числа запросов. Обеспечить корректность запросов парсинга интернет-магазинов не менее чем в 99,5% случаев от общего числа запросов. Время отклика на пользовательский запрос не должно превышать 10-ти секунд. В противном случае необходимо считать запрос неудавшимся, пользователю необходимо вывести сообщение об ошибке. При разработке информационной системы должны быть соблюдены нормы норм чередования умственной и физической нагрузки.

Пользователь должен иметь возможность зарегистрироваться на сайте для формирования личного кабинета. Также должна быть предусмотрена возможность смены пароля. Пользователь должен иметь возможность добавить товар в отслеживаемый. В случае, если он подписался на обновления через мессенджер Telegram, он должен получать уведомления об изменении цены на этот товар. Допускается до 10-ти одновременно отслеживаемых товаров на одного пользователя. В случае достижения лимита пользователю должно быть предложено удалить несколько других товаров из отслеживаемых. Пользователь должен иметь возможность отписаться от получения уведомлений об изменении цены на товар через мессенджер Telegram, через личный кабинет в системе, а также через карточку соответствующего товара с предварительной авторизацией по логину и паролю. Система должна предоставлять ссылку на карточку товара из интернет-магазина для каждой вещи. Веб-приложение должно формировать график динамики цены для каждого товара, представленного в системе. Максимальное количество предоставляемых записей из базы

данных для формирования графика не должно превышать 10-ти. Пользователь должен иметь возможность оценить товар и написать комментарий в карточке товара, который может быть опубликован после ручной модерации. Каждая вещь должна иметь краткий список ключевых характеристик на странице карточки товара. Пользователь должен иметь возможность удалить ранее опубликованный комментарий через карточку товара. Товары должны быть отсортированы по магазину, в котором они продаются, каждый магазин должен иметь свой список иерархических категорий.

Пароли пользователей должны быть защищены, необходимо использовать один из типов хеширования с высокой степенью криптографической стойкости. Это требование является общепринятым стандартом безопасности.

2.2. Проектирование базы данных и пользовательского интерфейса

Веб-приложение должно хранить внушительный объем информации, так как функционал подразумевает хранение данных о самих интернет-магазинах, категорий из этих магазинов, данных о товарах в каждой категории и историю цены для каждого товара. Также необходимо хранить пользовательские данные для входа, комментарии и оценки каждого пользователя, а также список его отслеживаемых товаров. ER диаграмма базы данных представлена в Приложении А.

Интерфейс программы должен быть интуитивно понятен пользователям системы. Веб-приложение должно использовать сочетающиеся цвета с умеренной контрастностью. Элементы меню должны быть интерактивны, подсвечиваться при наведении на них курсора, это даст пользователю понять, что элемент кликаемый и с ним можно взаимодействовать. Формы для ввода должны использовать валидацию входных данных, в случае если введенные данные не удовлетворяют

правилам ввода, поле должно подсвечиваться красным цветом. Рядом должно выводиться сообщение об ошибке. Кнопки и пункты меню должны сопровождаться иконками для интуитивного выбора действия пользователем. Иконки должны однозначно характеризовать действие в рамках тематики данного веб-приложения.

Веб-приложение использует технологию JPA Repository, которая позволяет воспользоваться готовым решением для выполнения стандартных GET, POST, PUT и DELETE запросов к базе данных. Благодаря использованию JPA Repository пропадает необходимость вручную прописывать SQL запросы к базе данных, но если такая необходимость все же появляется, то можно определить пользовательские запросы к базе данных, используя аннотацию @Query [5, с. 589]. Для каждого основного класса-сущности определяется свой репозиторий, который представляет собой интерфейс [2, с. 21]. После этого создается контроллер, которые обращается к репозиторио для передачи данных и получения ответа от него. JS библиотека использует расширение Vue Resource, которое позволяет указать куда передать сформированный клиентом запрос и вернуть ответ обратно клиенту. Веб-клиент отвечает за формирование запроса и обработку полученного ответа, после чего отображает изменения пользователю.

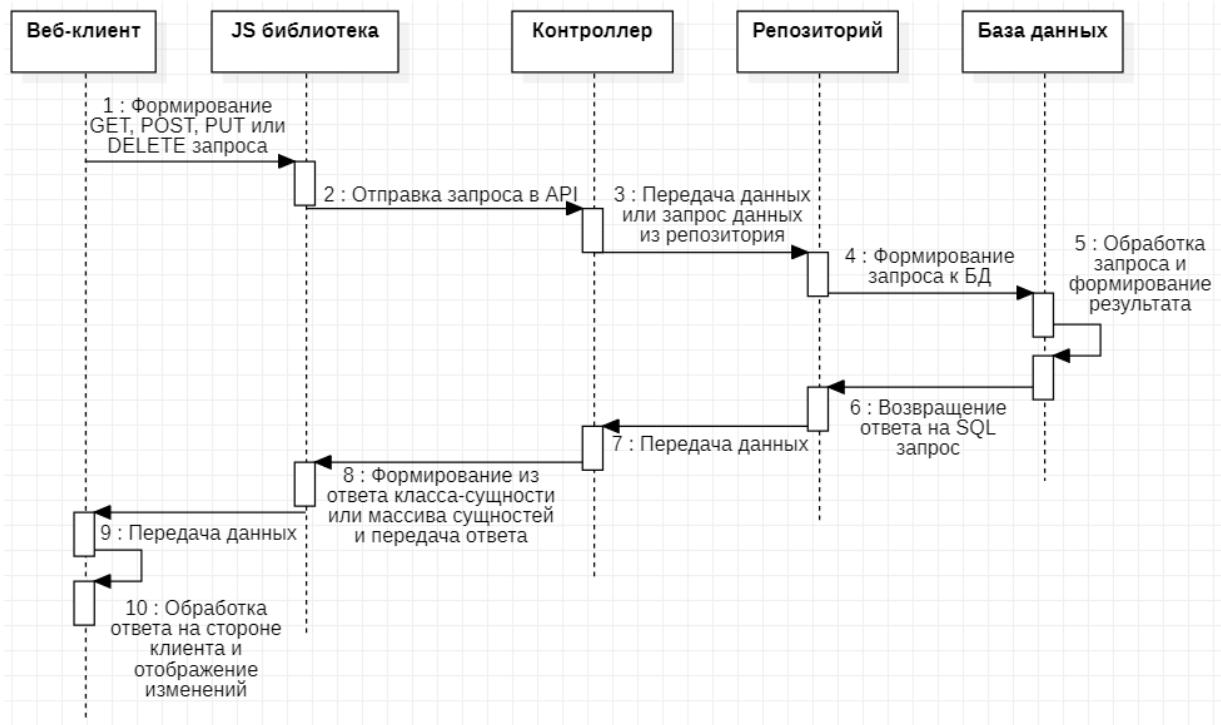


Рисунок 2.1. Диаграмма последовательности обработки запроса

3. Разработка многопользовательского веб-приложения

3.1. Реализация серверной части веб-приложения

Исходя из технического задания была написана серверная часть. Был сконфигурирован REST API сервер с помощью фреймворка Spring на языке программирования Java. Были сконфигурированы классы-сущности, представляющие все необходимые таблицы. Для каждого класса-сущности был создан JPA репозиторий, который предоставляет стандартные методы для выполнения запросов к базе данных: получение всех записей из таблицы, получение записи по первичному ключу, изменение записи по первичному ключу, удаление записи по первичному ключу, сохранение новой записи в базу данных. Некоторые репозитории были расширены с помощью дополнительных пользовательских запросов к базе данных. Были созданы контроллеры для обработки GET, POST, PUT и DELETE запросов со стороны клиента. При обработке запросов они обращаются к репозиториям для взаимодействия с базой данных. Использование фреймворков Spring и Hibernate позволяет быстро и просто сконфигурировать базу данных, а также реализовать большинство необходимых запросов для пользовательской части [7, с. 184].

Веб-приложение должно хранить большое количество информации: сведения о каждом представленном магазине, список категорий в каждом магазине, список товаров из каждой категории, информация об основных характеристиках каждого товара, ссылки на изображение товара, описание товара, оценка товара и количество отзывов в интернет-магазине, информация о цене каждого товара, пользовательские данные, комментарии и оценки пользователей, список отслеживаемых товаров, а также контактную информацию подписанных на оповещения пользователей. Для генерации базы данных и взаимодействия объектов были созданы классы-сущности,

после чего каждое поле было аннотировано с помощью Hibernate. Использовалось несколько типов аннотаций:

1. `@Id` – аннотация, позволяющая указать, что поле является первичным ключом.
2. `@GeneratedValue(strategy = GenerationType.IDENTITY)` – аннотация, позволяющая указать последовательный тип генерируемого значения первичного ключа [3, с. 357].
3. `@Column(name = "info", columnDefinition = "TEXT")` – аннотация, позволяющая указать название поля таблице, определить, может ли оно быть пустым при добавлении новой записи в БД, а также указать тип поля в БД при необходимости.
4. `@Entity` – аннотация, позволяющая указать, что класс является сущностью [4, с. 48].
5. `@Table(name = "prices")` – аннотация, позволяющая указать название таблицы в БД.
6. `@ManyToOne` – аннотация, позволяющая указать связь «многие-к-одному».
7. `@JoinColumn(name = "categories_low_id", nullable = false)` – аннотация, позволяющая сделать поле в БД внешним ключом. Благодаря этому можно указать ссылку на первичный ключ в другой таблицу, тем самым связав две разных таблицы.
8. `@OnDelete(action = OnDeleteAction.CASCADE)` – аннотация, позволяющая указать тип действия, которое произойдет при удалении первичного ключа, на который ссылается внешний ключ. В данном случае используется каскадное удаление записей, то есть все записи, связанные с удаленным первичным ключом в другой таблице, также будут удалены.

Каждая сущность описана ниже, а также указана ее основная функция в системе.

- Users – сущность, хранящая данные о почтовом адресе пользователя, пароле, ролях в системе, а также фото его профиля.
- Telegram – сущность, хранящая данные об уникальных идентификаторах пользователей, подписанных на уведомления об изменении цены в мессенджере Telegram.
- Shops – сущность, хранящая название интернет-магазина, информацию об интернет-магазине, логотип, а также ссылки на профили магазина в социальных сетях.
- Categories_high – верхний уровень иерархии списка категорий интернет-магазина, хранит название категории и ссылается на сущность Shops.
- Categories_middle – средний уровень иерархии списка категорий интернет-магазина, хранит название категории и ссылается на сущность Categories_high.
- Categories_low – низший уровень иерархии списка категорий интернет-магазина, хранит название категории и ссылается на сущность Categories_middle.
- Cities – сущность, хранящая названия городов.
- Items – сущность, хранящая информацию о названии товара, его рейтинг в интернет-магазине по пятибалльной шкале, количество комментариев, описание товара, оригинальную ссылку на страницу товара в интернет-магазине, а также ссылку на сущность Categories_low.
- Photos – сущность, хранящая ссылки на изображения каждого товара, ссылается на сущность Items.
- Prices – сущность, хранящая данные о цене на товар, времени парсинга цены, а также ссылки на сущности Items и Cities.
- Categories_of_Specifications – сущность, хранящая информацию о заголовке описания товара, ссылается на сущность Items.

- Specifications – сущность, хранящая название характеристики и ее значение, ссылается на сущность Categories_of_Specifications.
- Favorite_Items – сущность, хранящая данные об отслеживаемых пользователем товарах, ссылается на сущности Users и Items.
- Review – сущность, хранящая пользовательские комментарии и оценки о товаре, ссылается на сущности Users и Items.

Также в приложении определены некоторые дополнительные сущности, необходимые для работы серверной части с клиентскими запросами, они не аннотированы, так как используются в качестве дополнительных сущностей и не должны сохранять информацию в БД напрямую. Для них был определен дополнительный пакет extra, который располагается в пакете entity.

- ChangePasswordEntity – вспомогательная сущность, необходимая в процессе смены пароля пользователем. В ней определены поля для нового пароля, старого пароля и подтверждения нового пароля.
- RegistrationUserEntity – вспомогательная сущность, необходимая в процессе регистрации нового пользователя. В ней определены поля для почтового адреса и пароля.
- ReviewEntity – вспомогательная сущность, необходимая в процессе публикации отзыва и оценки о товаре пользователем. В ней определены поля для текста комментария, рейтинга. Вспомогательный класс ссылается на сущность Items.
- StatusEnum – вспомогательный класс перечисляемого типа, используемый для хранения возможных статусов выполнения запросов и операций. Включает в себя 4 различных статуса:

1. Success – успешное выполнение операции;
2. UserAlreadyExists – пользователь с таким почтовым адресом уже зарегистрирован в системе;
3. Error – произошла ошибка в процессе выполнения операции;

4. NotAuthorized – пользователь не авторизирован.

Система предполагает большое количество таблиц, связанных между собой. Для генерации стандартных запросов использовалась спецификация JPA. Были созданы отдельные репозитории для каждого из классов. Благодаря этому для каждого класса сущности автоматически были реализованы операции получения всех записей, получения определенной записи, удаления определенной записи, изменения определенной записи и сохранения новой записи в БД. Однако некоторым компонентам необходимы более сложные структуры данных, которые нельзя быстро и просто получить стандартными запросами. В этом случае JPA позволяет определить расширенные запросы с помощью аннотации @Query.

- UsersRepositories – репозиторий для сущности Users. В этом репозитории определен расширенный запрос findUserByEmail, который позволяет найти пользователя по его почтовому адресу.
- TelegramRepositories – репозиторий для сущности Telegram. В этом репозитории определен расширенный запрос findTelegramByUserId, который позволяет найти уникальный идентификатор пользователя по его почтовому адресу.
- SpecificationsRepositories – репозиторий для сущности Specifications. В этом репозитории определен расширенный запрос findSpecificationsByCategories_of_SpecificationsId, возвращающий только те записи, которые принадлежат определенной категории по ее первичному ключу.
- ShopsRepositories – репозиторий для сущности Shops. В этом репозитории определен расширенный запрос findLastShop, который возвращает массив объектов Shops, отсортированных в порядке убывания по первичному ключу.

- ReviewRepositories – репозиторий для сущности Review. В этом репозитории определен расширенный запрос findReviewsById, позволяющий получить все отзывы к определенному товару.
- PricesRepositories – репозиторий для сущности Prices. В этом репозитории определен расширенный запрос findPricesById для получения только тех цен, которые принадлежат определенному товару.
- PhotosRepositories – репозиторий для сущности Photos. В этом репозитории определен расширенный запрос findPhotosById для получения только тех изображений, которые принадлежат определенному товару, отсортированные по возрастанию первичного ключа.
- ItemsRepositories – репозиторий для сущности Items. В этом репозитории определен расширенный запрос findItemsByShopId, который возвращает все товары, принадлежащие определенному магазину. Также был определен запрос findItemsByLink, который возвращает товар по его
 - Favorite_ItemsRepositories – репозиторий для сущности Favorite_Items. В этом репозитории определен расширенный запрос findItemsByUserId, который позволяет получить все отслеживаемые пользователем товары. Также определен запрос isFavoriteItem, который позволяет узнать, является ли предмет отслеживаемым у данного пользователя.
 - Categories_of_SpecificationsRepositories – репозиторий для сущности Categories_of_Specifications. В этом репозитории определен расширенный запрос findCategories_of_SpecificationsById, который возвращает все заголовки характеристик для определенного товара.
 - Categories_middleRepositories – репозиторий для сущности Categories_middle. В этом репозитории определен расширенный запрос findMiddleCategoriesByHighCategory, который позволяет получить все категории среднего уровня по родительской категории верхнего уровня.

- Categories_lowRepositories – репозиторий для сущности Categories_low. В этом репозитории определен расширенный запрос findLowCategoriesByMiddleCategory, который позволяет получить все категории нижнего уровня по родительской категории среднего уровня.

- Categories_highRepositories – репозиторий для сущности Categories_high. В этом репозитории определен расширенный запрос findCategoriesByShopId, который позволяет получить все категории верхнего уровня по выбранному интернет-магазину.

Для обработки GET, POST, PUT и DELETE запросов были написаны контроллеры. Каждый контроллер отмечен аннотацией @RestController. Контроллер содержит конструктор и приватные поля для репозиториев, описанных ранее. Каждое поле отмечено аннотацией @Autowired, благодаря этому при создании экземпляра класса в приватное поле будет инициализирован необходимый репозиторий, после чего методы класса смогут взаимодействовать с ним для выполнение стандартных или расширенных запросов к базе данных. Для привязки метода к запросу используются специальные аннотации:

1. @GetMapping – аннотация для приема GET запросов, обычно с помощью запросов такого типа запрашивается информация [18];
2. @PostMapping – аннотация для приема POST запросов. Во входные параметры с помощью аннотации @RequestBody может быть передано тело запроса. Обычно такой тип запроса используется для добавления новой записи в базу данных;
3. @PutMapping – аннотация для приема PUT запросов. Во входные параметры может быть передано тело запроса и уникальный идентификатор, обозначаемый аннотацией @PathVariable. Обычно в качестве идентификатора используется первичный ключ объекта. В большинстве случаев такой тип запроса используется для изменения записи в базе данных;

4. @DeleteMapping – аннотация для приема DELETE запросов. Во входные параметры может быть передан уникальный идентификатор. Обычно используется первичный ключ объекта. Обычно такой тип запроса используется для удаления записей из базы данных.

- UsersController – контроллер, отвечающий за процесс регистрации, для этого в нем определен метод registration, возвращающий объект StatusEnum. В процессе регистрации введенный почтовый адрес проверяется на уникальность, после чего введенный пароль шифруется алгоритмом BCrypt, который использует соль при генерации хеша, после чего информация сохраняется в базу данных [6, с. 432].
- TelegramController – контроллер, отвечающий за подписку пользователей на уведомления о смене цены на товары из списка отслеживаемых. Позволяет сохранить уникальный идентификатор пользователя в мессенджере Telegram, а также отписаться от уведомлений и удалить данные об уникальном идентификаторе.
- SpecificationsController – контроллер, отвечающий за сохранение, изменение, удаление и получение информации о характеристиках товара. Также определен метод, позволяющий получить всю информацию о характеристиках для выбранного товара.
- ShopsController – контроллер, отвечающий за сохранение, изменение, удаление и получение информации об интернет-магазине. В нем также определен дополнительный метод, позволяющий получить самый новый интернет-магазин, находящийся в базе данных. Для этого применяется фильтрация по первичному ключу.
- ReviewController – контроллер, который отвечает за получение всех отзывов о конкретном товаре.
- PricesController – контроллер, отвечающий за сохранение, изменение, удаление и получение информации о цене на товар. В нем также

определен дополнительный метод, позволяющий получить список цен на товар за все время.

– PhotosController – контроллер, отвечающий за получение основного и дополнительных изображений о товаре. Также в нем определена переменная default_photo, которая хранит ссылку на стандартное изображение, если в интернет-магазине не представлены фотографии товара.

– ItemsController – контроллер, отвечающий за сохранение, изменение, удаление, получение информации о товарах в интернет-магазинах и за уведомление пользователей в Telegram при изменении цены на отслеживаемый ими товар. В нем также определен метод для получения всех товаров из выбранных категорий верхнего, среднего и нижнего уровней. Был определен метод для получения товара по оригинальной ссылке в интернет-магазине. Он необходим для получения информации о товаре из базы данных

При обработке данных от парсера происходит сверка всех отслеживаемых пользователями предметов с текущим товаром. Если пользователь поместил этот товар в список отслеживаемых и подключил уведомления через Telegram, ему будет отправлено уведомление об изменении цены на этот товар.

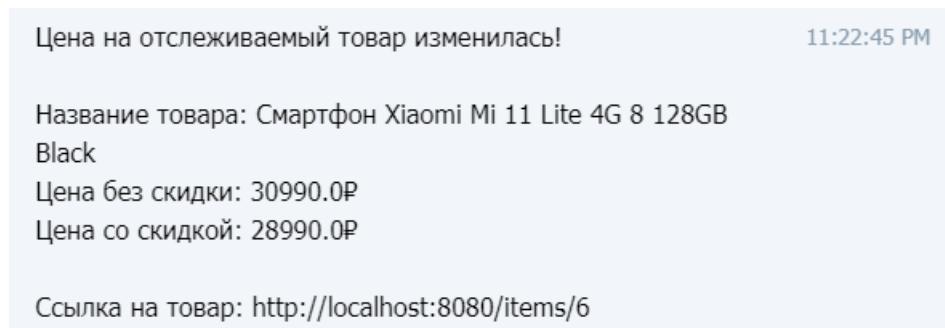


Рисунок 3.1. Уведомление об изменении цены в Telegram

– Favorite_ItemsController – контроллер, отвечающий за получение всех отслеживаемых пользователем товаров, а также за добавление товара в список отслеживаемых и удаление его оттуда. Дополнительно определен метод, позволяющий определить, находится ли выбранный товар в списке

отслеживаемых у пользователя. Для получения информации о пользователе используется класс Authentication, который хранит информацию об авторизации пользователя в системе. Из него запрашивает почтовый адрес пользователя, после чего передается в репозиторий для поиска пользователя по данному почтовому адресу.

- Categories_of_SpecificationsController – контроллер, отвечающий за изменение, сохранение, получение и удаление информации о заголовках описания товара. В нем также определен дополнительный метод, позволяющий получить список всех заголовков для выбранного товара.
- Categories_by_ShopController – контроллер, отвечающий за получение всех категорий определенного интернет-магазина. В нем также определен дополнительный класс Category, выполняющий роль адаптера между контроллером и пользовательской частью.
- AuthController – контроллер, отвечающий за проверку авторизации пользователя, смену пароля пользователя, а также за поиск пользователя по его почтовому адресу.

Проверка на авторизацию пользователя происходит при запросе данных авторизации из класса Authentication, который хранит почтовый адрес пользователя. Если вместо почтового адреса метод получает anonymousUser, то это значит, что пользователь не авторизирован.

Для импорта библиотек в проект используется популярный фреймворк Maven. Основным файлом конфигурации является pom.xml. В нем хранятся все названия подключаемых библиотек, а также их версии [16]. В случае переноса проекта, файл конфигурации Maven будет прочитан системой, после чего необходимые модули будут скачаны, распакованы и установлены автоматически. Это также упрощает контроль версий подключаемых библиотек со стороны разработчика. Для импорта расширений отлично подходит веб-ресурс Maven Repository, который позволяет найти

необходимую библиотеку и предоставляет код в формате XML для ее импорта в проект.

В веб-приложении используется библиотека Spring Boot, которая необходима для инициализации стандартного Spring проекта. Spring Security используется для реализации алгоритмов авторизации и регистрации, благодаря ему можно разграничить доступ пользователей с различным уровнем привилегий к частям веб-приложения, а также для построения стандартных методов защиты пользователей [20]. Библиотека PostgreSQL JDBC Driver нужна для обеспечения соединения с базой данных, а также выполнения SQL запросов. В pom.xml перечислены все необходимые модули с нужными версиями. Код представлен в Приложении Б.

3.2. Реализация пользовательской части информационной системы

Для написание пользовательской части использовался фреймворк Vue.js. Основным файлом конфигурации является main.js, в котором импортируются дополнительные компоненты, графические фреймворки BootstrapVue и Vuetify, а также определяются глобальные переменные [9].

```
// views/main.js
import { BootstrapVue, IconsPlugin } from 'bootstrap-vue'
import vuetify from "./plugins/vuetify"
import Lingallery from 'lingallery';
import TrendChart from "vue-trend-chart";
import VueSidebarMenu from 'vue-sidebar-menu'
import 'vue-sidebar-menu/dist/vue-sidebar-menu.css'

Vue.use(BootstrapVue)
Vue.use(IconsPlugin)
Vue.use(TrendChart);
Vue.use(VueSidebarMenu)
Vue.component('lingallery', Lingallery);

new Vue({
  router,
  vuetify,
```

```
    render: h => h(App)
}).$mount('#app')
```

Для навигации по веб-страницам приложения используется Router.js, который хранит в себе все доступные пути, а также сопоставляет компоненты с запрашиваемыми адресами [12].

```
// views/router.js
export default new Router({
  mode: 'history',
  routes: [
    {
      path: '/', component: DefaultPage, name: 'DefaultPage'
    },
    {
      path: '/items/:item_id', component: ItemPage, name: 'ItemPage', props: true
    },
    {
      path: '/shops/:shop_id/categories', component: ItemsPage, name: 'ItemsPage', props: route => ({params: route.params, high_id: route.query.high, middle_id: route.query.middle, low_id: route.query.low})
    },
  ]
})
```

Важной особенностью Vue.js является возможность разделения частей веб-страницы на отдельные компоненты [8, с. 246]. Это позволяет логически разделить код на несколько разных файлов, что упрощает работу с ним, после чего он будет преобразован в одну страницу автоматически. Поэтому каждая страница была разделена на составные компоненты, которые были помещены в пакет components, а файлы в пакете pages выступают в роли контейнеров для компонентов.

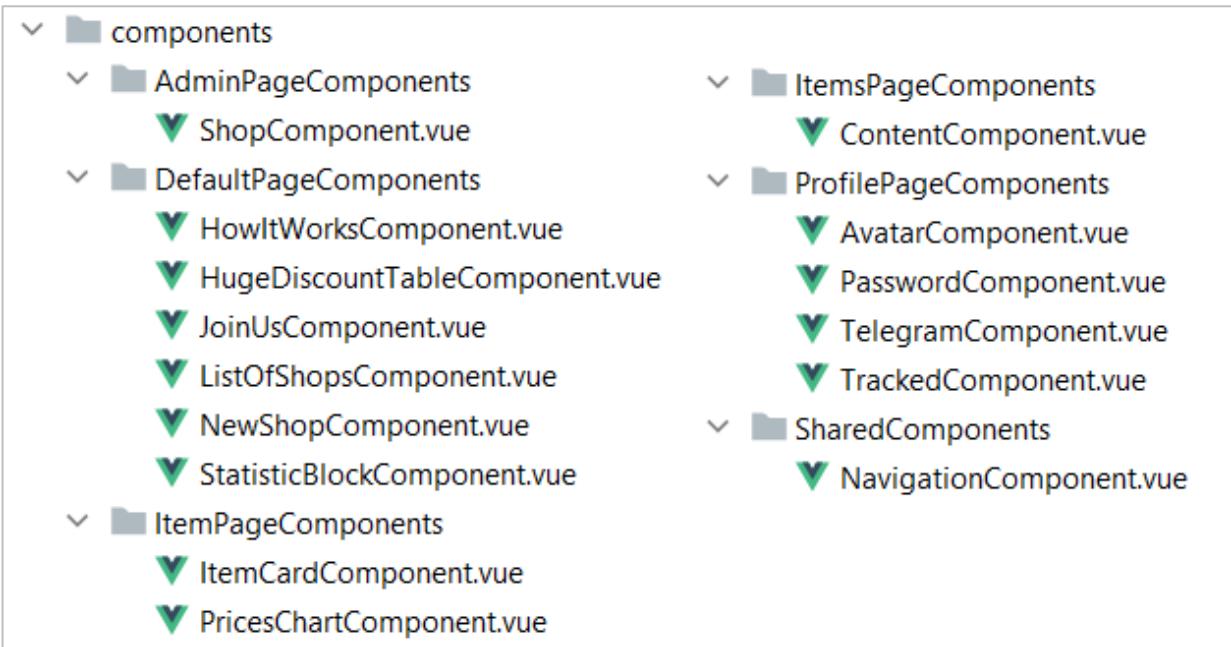


Рисунок 3.2. Компоненты веб-страниц

Для получения данных и формирования запросов используется библиотека `VueResource`, которая позволяет быстро и просто сформировать POST, GET, PUT или DELETE запрос, а после отправить его на конечную точку API. Для каждого контроллера была сгенерирована своя библиотека API, которая представляет собой JS файл. В нем определены пути для обращения к контроллерам, а также указан формат и тип передаваемых данных.

```

// api/shops.js
import Vue from 'vue'
import VueResource from 'vue-resource'
Vue.use(VueResource)
const shop = Vue.resource('/api/shops/{id}');
const lastShop = Vue.resource('/api/shops/last');
export default {
  get: () => shop.get(),
  getLastShop: () => lastShop.get(),
  getOne: id => shop.get({id}),
  add: (body) => shop.save(body),
  update: (id, body) => shop.update({id}, body),
  remove: id => shop.remove({id}),
}

```

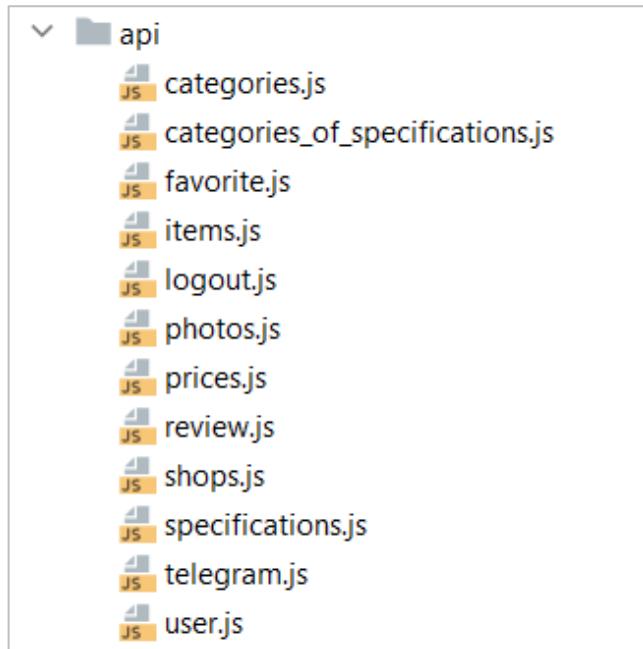


Рисунок 3.3. Список API библиотек

Главная страница приложения содержит несколько основных элементов: компонент, отвечающий за отображение нового магазина и части его товаров, компонент для отображения наибольших скидок по всем товарам, компонент для отображения статистики по всем интернет-магазинам и товарам, представленным в системе, а также компонент, отвечающий за отображение каждого интернет-магазина в отдельности.

Изображение	Название	Текущая стоимость	Изменение	График цены	Просмотр
	Электрочайник Philips HD9339/80	3 290 ₽	-34.07 %		ПОСМОТРЕТЬ >
	Смартфон Xiaomi Mi 11 Lite 4G	28 990 ₽	-6.45 %		ПОСМОТРЕТЬ >

Рисунок 3.4. Главная страница информационной системы

Компонент, отвечающий за отображение самых больших скидок, получает информацию о всех товарах, представленных в системе. Помимо вывода основных сведений о товарах, таких как изображение товара, название и текущая стоимость, рассчитывается размер скидки, выраженный в процентах по отношению к первоначальной цене до скидки, а также строится график цены, позволяющий наглядно оценить динамику колебания цены.

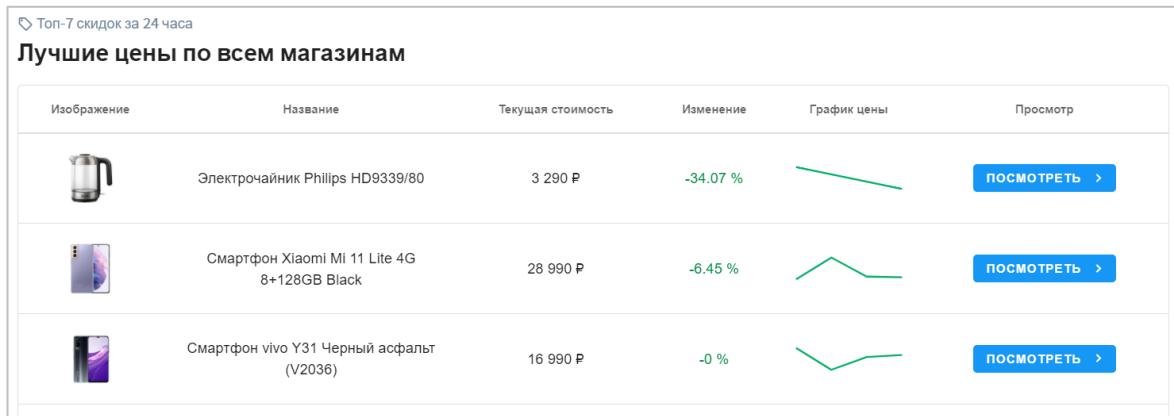


Рисунок 3.5. Компонент, отображающий товары с наибольшими скидками

На главной странице реализован компонент, выводящий все магазины, присутствующие в системе. В каждый карточке магазина выводится информация о названии магазина, его описание, логотип и несколько товаров, представленных в этом интернет-магазине.

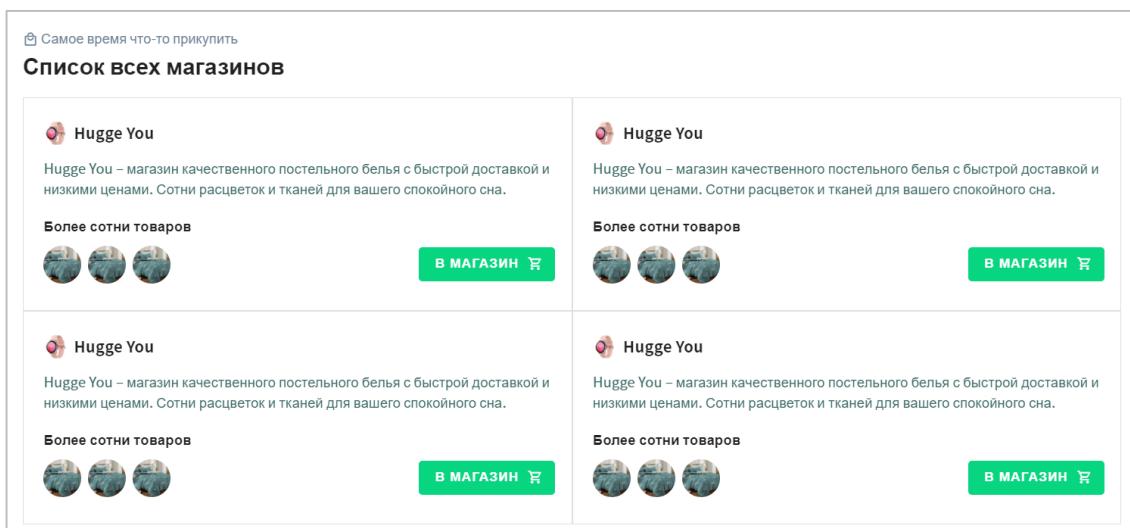


Рисунок 3.6. Компонент, выводящий все магазины, присутствующие в системе

При переходе в определенный интернет-магазин отображается страница со списком всех товаров, представленных в этом магазине, а также список всех категорий, определенных в этом интернет-магазине. Список категорий находится в левой части страницы и представляет собой иерархический список, в котором дочерние элементы будут показаны только при нажатии по родительской категории. Отображаемый список товара будет динамически изменяться, на странице будут показаны только те товары, который принадлежат выбранной пользователем категории. Для построения иерархического списка категорий использовалась библиотека Vue Sidebar Menu, которая принимает список с дочерними элементами, после чего отображает их в такой же последовательности [21].

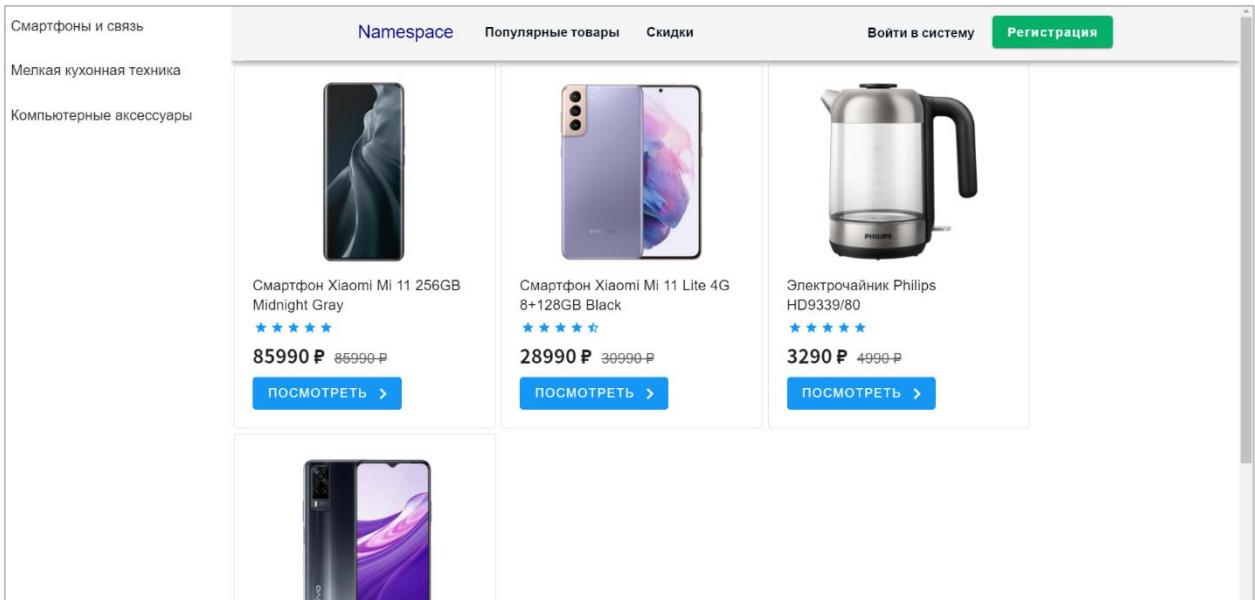


Рисунок 3.7. Страница интернет-магазина

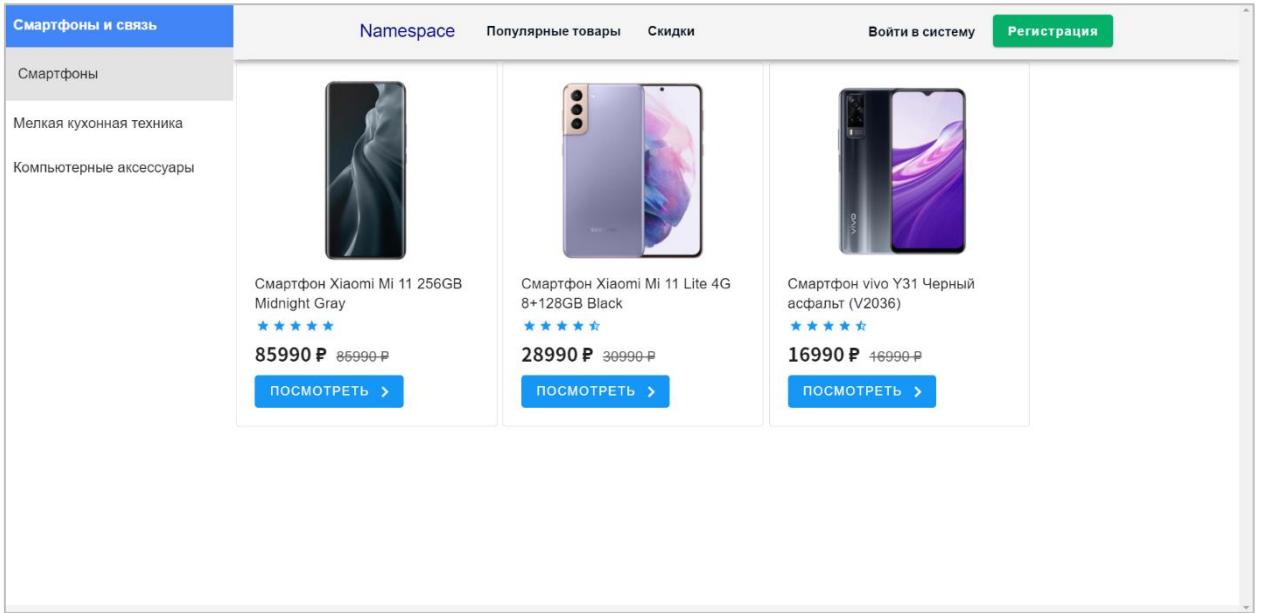


Рисунок 3.8. Страница интернет-магазина с выбранной категорией

При выборе определенного товара отображается страница с основной информацией об этом предмете. Она состоит из навигационного меню, в котором последовательно представлены категории, к которым принадлежит данный товар. Ниже располагается секция с названием товара, а также пользовательской оценкой в интернет-магазине по пятибалльной шкале и количеством отзывов. Рядом также находится кнопка, позволяющая добавить товар в список отслеживаемых. Для этого действия необходима предварительная авторизация. Если пользователь не авторизирован в системе, при нажатии кнопки он будет автоматически перенаправлен на страницу авторизации. Слева находится галерея изображений товара, для нее использовалась библиотека Lingallery. Была изучена документация к ней. Библиотека позволяет переключаться между всеми изображениями, а также выводит выбранное изображение в увеличенном формате [15]. В правой части располагается секция с указанием текущей стоимости товара в интернет-магазине, предыдущей цены до скидки, а также автоматически высчитывается процент скидки на данный товар. Ниже расположена кнопка

для перехода по оригинальной ссылке товара. Также в этой секции находится краткий список основных характеристик товара.

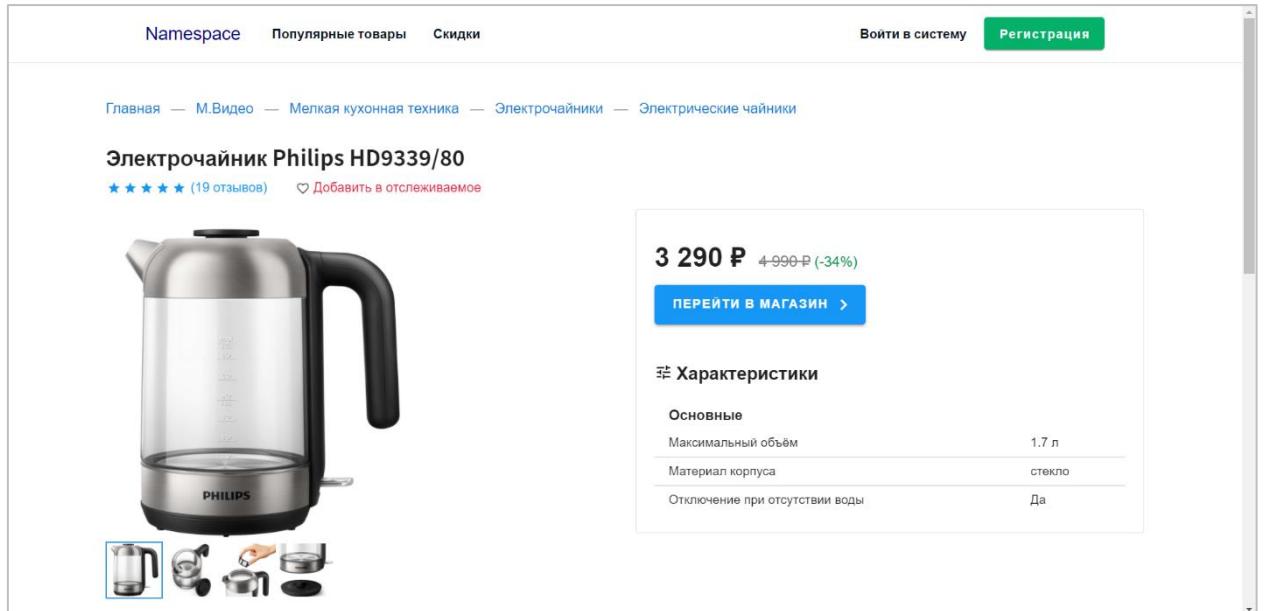


Рисунок 3.9. Страница товара

Ниже основной информации располагается график динамики цены товара. Для построения графика использовалась дополнительная библиотека Vue Trend Chart, которая представляет собой набор из нескольких компонентов и JS файлов, обрабатывающих логику при построении графика цены [22]. По оси абсцисс располагаются даты, на которые была актуальна указанная цена, по оси ординат указаны цены в рублях. Точкой на графике обозначается актуальная цена на указанное число.

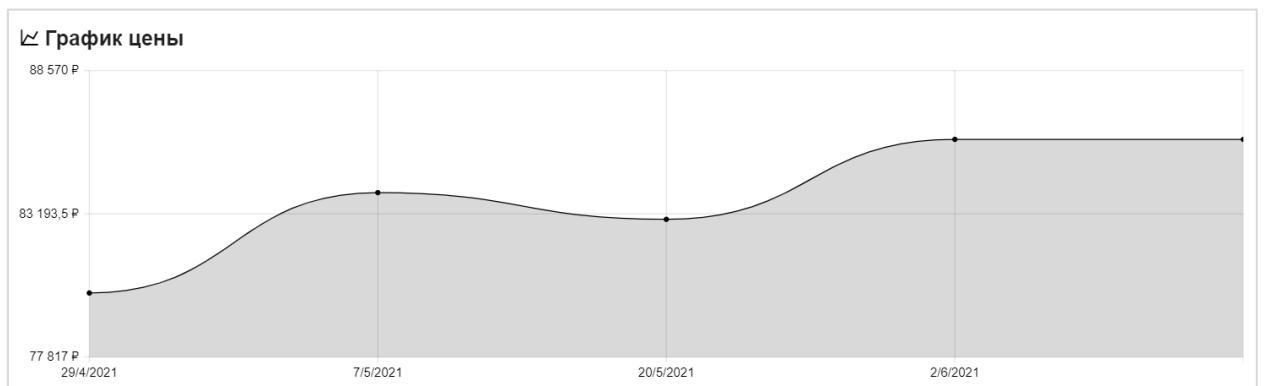


Рисунок 3.10. График динамики цены товара

Под графиком находится секция с описанием товара из интернет-магазина. Ниже расположена форма для публикации комментария о товаре, а также его оценки по пятибалльной шкале пользователями системы. Для оставления отзыва необходимо обязательно написать текст комментария и оценить товар. Без выполнения хотя бы одного действия отзыв опубликован не будет, а пользователь увидит сообщение об ошибке. Такой подход наглядно показывает пользователю системы, где он ошибся и что необходимо исправить. Для публикации отзыва необходима предварительная авторизация. Если пользователь не авторизирован в системе, он автоматически будет перенаправлен к форме авторизации. Максимальная длина отзыва не должна превышать трехсот символов, в противном случае пользователю будет показано сообщение об ошибке, а счетчик символов в нижнем левом углу будет отмечен красным цветом. После отправки отзыва он будет автоматически опубликован в карточке выбранного товара.

① Информация

Вы можете купить Смартфон Xiaomi Mi 11 256GB Midnight Gray в магазинах М.Видео по доступной цене. Смартфон Xiaomi Mi 11 256GB Midnight Gray: описание, фото, характеристики, отзывы покупателей, инструкция и аксессуары.

✉ Отзывы

Напишите свое мнение о товаре

Пользуюсь около 3-4 месяцев, брал за 7 т.р со всеми скидками и бонусами. Среди бюджетников один из лучших вариантов. После покупки сразу же снес почти все встроенные приложения, благо это делается относительно легко. Автономности устройства хватает на 1-2 дня при использовании соц. сетей, прослушивания музыки и приложений, в игры не играю.

Отзыв не должен превышать 300 символов

341

☆ ☆ ☆ ☆ ☆ ОПУБЛИКОВАТЬ 

Необходимо оценить товар

Рисунок 3.11. Секция с информацией о товаре и форма для отправки отзыва

Ниже располагается список всех отзывов пользователей системы о выбранном товаре, а также пользовательские оценки по пятибалльной шкале. В качестве имени пользователя в системе используется почтовый адрес, однако он является чувствительной информацией, поэтому при публикации отзыва часть логина скрывается специальными символами.

	blo**@mail.ru
	★★★★★
Чайник работает тихо, по сравнению с предыдущим, тот когда закипал шумел как трактор. А этот тихо кипит, но не беззвучно. Из плюсов для себя могу отметить съёмную крышку, у двух других чайников ломались именно крышки. В общем чайник очень понравился.	
<hr/>	
	mau**@mail.ru
	★★★★★
Вместительный чайник на 1,7 л. Удобный носик с фильтром. Он не такой шумный, как наш предыдущий чайник. Мне очень понравилась мерная шкала, на которой нарисованы чашечки (сразу видно на сколько гостей хватит налить чая). Хорошо вписывается в интерьер нашей кухни.	
<hr/>	
	bro**@bk.ru
	★★★★★
Всегда мечтал о прозрачном чайнике с подсветкой и наконец-то приобрел такой. Чайник очень удобен в использовании - не громоздкий, довольно лёгкий, удобное расположение кнопки включения и крышки - она легко открывается и закрывается, не нужно прикладывать никаких усилий.	

Рисунок 3.12. Секция с отзывами пользователей

Для авторизации пользователя необходимо ввести валидную пару почтового адреса и пароля. Оба поля используют предварительную проверку вводимых данных. Если формат данных не совпадает с допустимым пользователю будет показана ошибка.

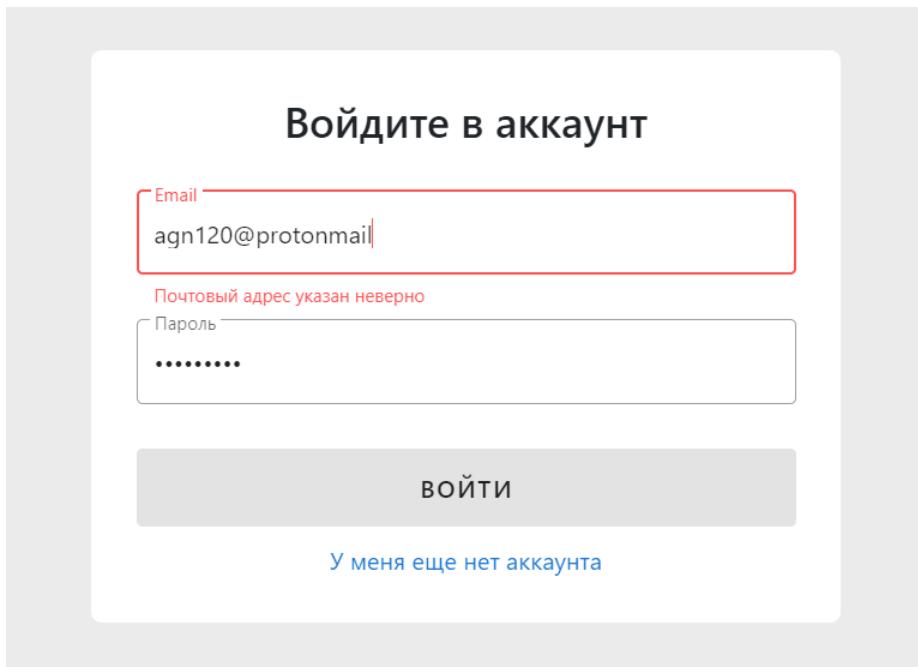


Рисунок 3.13. Страница авторизации пользователя

Если пользователь еще не имеет аккаунта в системе, он может создать аккаунт, перейдя на страницу регистрации. Однако его почтовый адрес не должен быть использован в системе ранее. Если почтовый адрес уже занят, пользователь увидит сообщение об ошибке. Поля также используют предварительную валидацию данных. Введенный пользователем пароль не должен быть короче восьми символов.

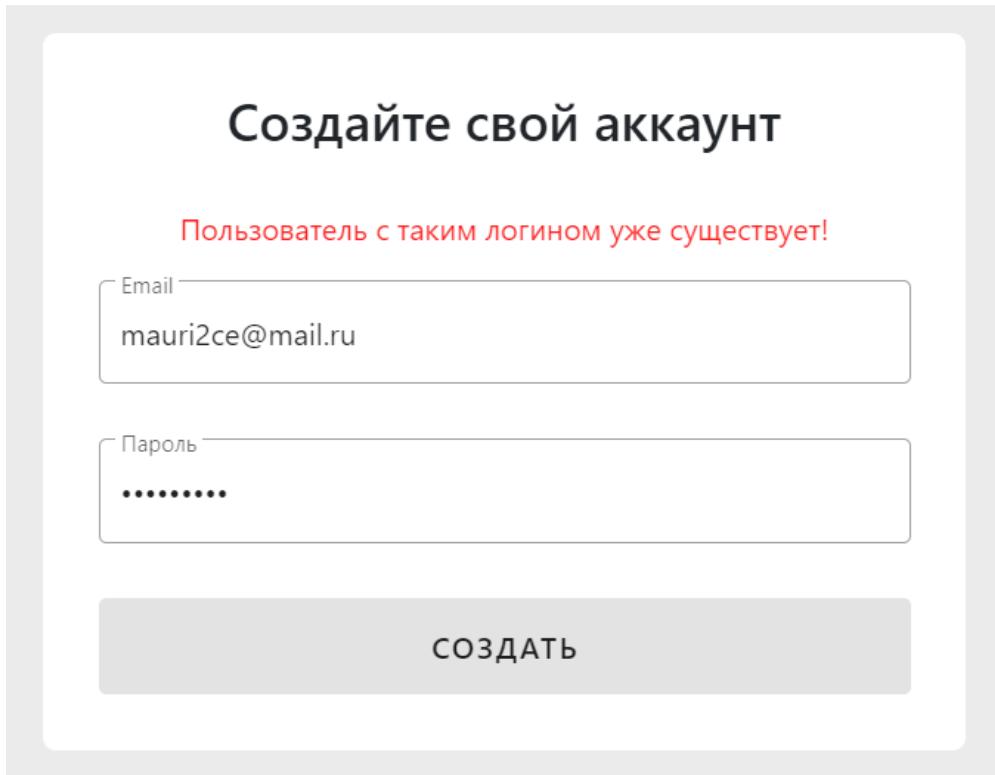


Рисунок 3.14. Страница регистрации пользователя

У каждого пользователя при регистрации создается личный кабинет, в котором он может сменить пароль, установить изображение профиля, просмотреть список отслеживаемых товаров, а также отредактировать его и подключить профиль мессенджера Telegram для получения уведомлений об изменении цены отслеживаемых товаров.

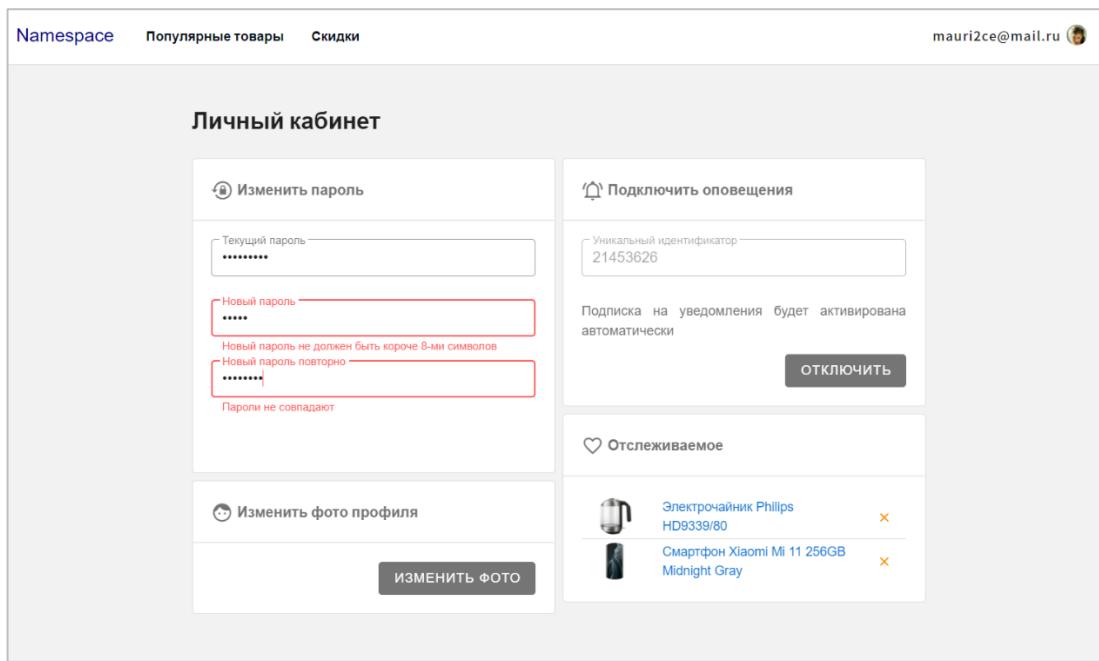


Рисунок 3.15. Страница регистрации пользователя

При смене пароля пользователь должен указать старый пароль, а также ввести новый и его подтверждение. Если старый пароль будет введен неверно, пользователь получит сообщение об ошибке. Также к полям нового пароля и подтверждения пароля применяется предварительная валидация, которая не позволит установить новый пароль короче восьми символов, а подтверждение пароля обязательно должно совпадать с самим паролем, иначе пользователь получит уведомление об ошибке.

A detailed view of the password change form. At the top is a header 'Изменить пароль'. Below it, an error message 'Текущий пароль не верен' (Current password is incorrect) is displayed above the 'Текущий пароль' (Current password) input field. A placeholder text 'Пожалуйста, введите свой текущий пароль' (Please enter your current password) is shown above the field. Below the current password field are fields for 'Новый пароль' (New password) and 'Новый пароль повторно' (New password repeat), both containing placeholder text '*****'.

Рисунок 3.16. Форма смены пароля с указанием ошибки

При нажатии на кнопку смены изображение профиля будет открыто модальное окно проводника Windows, позволяющее выбрать желаемое изображение, после чего оно будет установлено в качестве фотографии профиля пользователя.

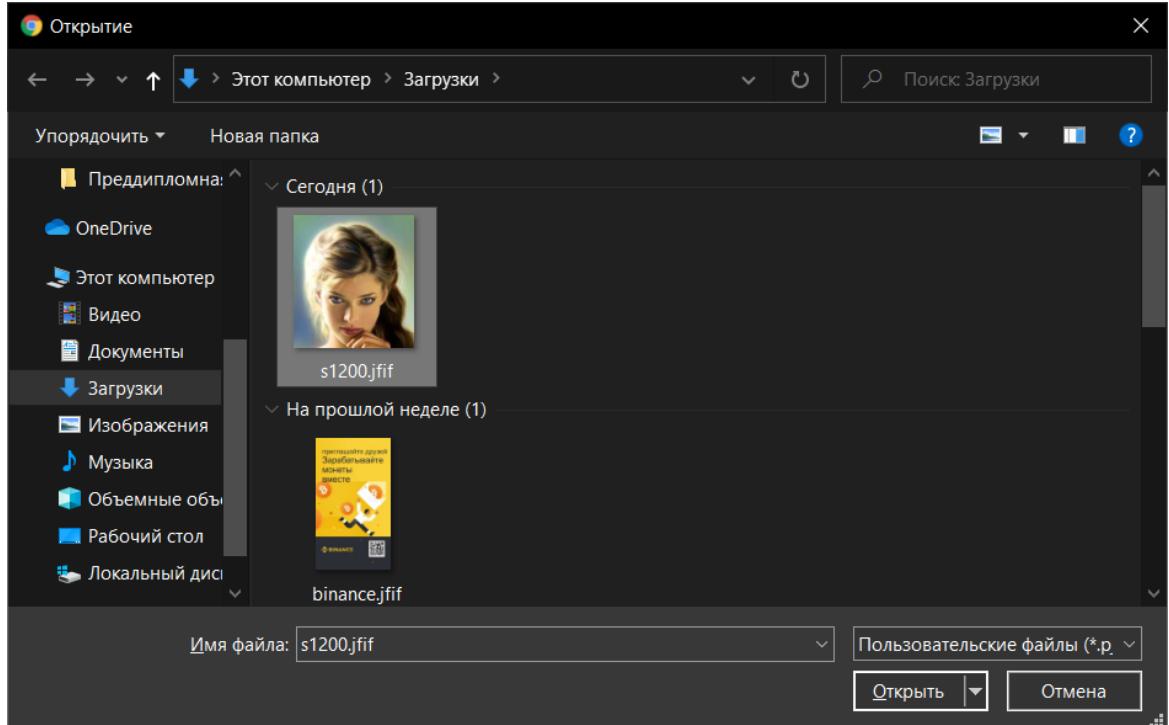


Рисунок 3.17. Модальное окно выбора изображения для профиля

В правом верхнем углу находится выпадающее меню, позволяющее перейти в личный кабинет пользователя или выйти из своего профиля.

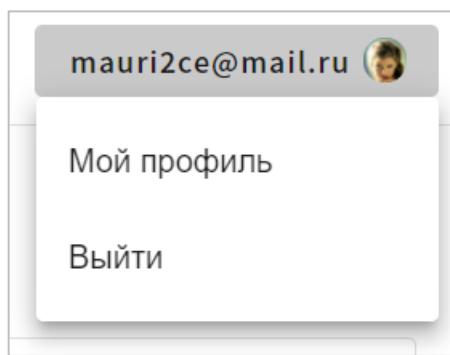


Рисунок 3.18. Выпадающее меню

Для администратора реализована админ-панель, в которой он может добавить в систему новый интернет-магазин. В выпадающем меню для

администратора доступен пункт «Админ-панель», при нажатии по нему будет загружена таблица уже имеющихся в системе магазинов, после чего их можно будет отредактировать, удалить или добавить в систему новый интернет-магазин.

The screenshot shows the Admin Panel interface. At the top, there are tabs: 'Namespace', 'Популярные товары' (Popular products), and 'Скидки' (Discounts). Below these tabs is a table with columns: 'Id', 'Название' (Name), 'Информация' (Information), and 'Ссылки' (Links). There are two rows in the table:

Id	Название	Информация	Ссылки
5	Эльдорадо	Эльдорадо - интернет-магазин электроники, цифровой и бытовой техники, выгодные цены, доставка по Москве и регионам	<ul style="list-style-type: none">@YouTubeInstagramVK
1	M.Video	M.Video - описание	<ul style="list-style-type: none">@YouTubeInstagramVK

To the right of the table, a modal window is open for the store with Id 5, titled 'admin@admin.ru'. It contains sections for 'Мой профиль' (My profile) with fields for 'Изменить' (Edit) and 'Удалить' (Delete), and a 'Выход' (Logout) button. Below this is a list of links with edit and delete icons. At the bottom of the modal is a green 'Добавить новый магазин' (Add new store) button.

Рисунок 3.19. Админ-панель

При нажатии на кнопку редактирования пользователь увидит модальное окно, поля в котором будут автоматически заполнены данными из строки таблицы. После их изменения достаточно нажать на кнопку сохранения, данные будут обновлены автоматически. Если нажать на кнопку отмены, никакие данные сохранены не будут, изменения не сохранятся.

Изменить данные магазина

Название
Эльдорадо

Информация
Эльдорадо - интернет-магазин электроники, цифровой и бытовой техники, выгодные цены, доставка по Москве и регионам

Twitter
https://twitter.com/eldorado_stores

YouTube
<https://www.youtube.com/user/eldoradovideo>

Instagram
https://www.instagram.com/eldorado_ru/

VK
https://vk.com/eldorado_stores

ОТМЕНА **СОХРАНИТЬ**

Рисунок 3.20. Форма редактирования данных интернет-магазина

При нажатии на кнопку удаления выбранный интернет-магазин будет выведен из системы, данные об оставшихся интернет-магазинах обновятся автоматически и появятся на странице.

3.3. Разработка модулей для автоматизированного мониторинга цен

Были написаны скрипты для парсинга страниц интернет-магазинов на языке программирования Python. Взаимодействие скриптов и REST API сервера на Java осуществляется с помощью отправки GET и POST запросов [1, с. 160]. Для получения кода страниц интернет-магазинов используется библиотека Requests, позволяющая формировать запросы [11]. Все данные с REST API сервера передаются в формате JSON, поэтому для быстрой обработки используется модуль Pykson. Он позволяет сопоставить имена

переменных в классах с именами переменных в ответе от REST API сервера, тем самым инициализировав ту же структура классов, которая была определена на REST API сервере, с теми же данными [17]. Для создания аналогичной структуры классов также использовался модуль Pykson, а именно его расширения JsonObject, IntegerField, StringField, FloatField, ObjectField, TimestampSecondsField.

1. JsonObject – входной объект типа JSON, который будет распарсен при обработке данных;
2. IntegerField – тип данных для инициализации поля числом;
3. StringField – тип данных, для инициализации поля строкой;
4. FloatField – тип данных, для инициализации поля вещественным числом;
5. ObjectField – тип данных, для инициализации поля объектом пользовательского типа;
6. TimestampSecondsField – тип данных для инициализации поля типом Timestamp.

```
// mvideo_parser.py
class Categories_low(JsonObject):
    id = IntegerField()
    name = StringField()
    categories_middle_id = ObjectField(Categories_middle)

class Items(JsonObject):
    id = IntegerField()
    title = StringField()
    comments = IntegerField()
    info = StringField()
    link = StringField()
    rating = FloatField()
    categories_low_id = ObjectField(Categories_low)
```

Также в скрипте были определены заголовки запросов, имитирующие реальный браузер. Этот подход позволяет избежать дополнительных проверок со стороны сайта при парсинге его страниц.

Изначально скрипт получает все предметы, присутствующие в системе и принадлежащие определенному интернет-магазину.

```
// mvideo_parser.py
def getItemsByShopId():
    result =
    requests.get("http://localhost:8080/api/items/byshop/" +
    str(SHOP_ID) +
    "/high/null/middle/null/low/null").content.decode("utf-8")
    items = Pykson().from_json(result, Items)
    print("Загружено товаров: " + str(len(items)))
    return items
```

После чего запускается цикл, проверяющий каждый из товаров последовательно. После окончания проверки всех товаров алгоритм приступает заданное количество секунд и повторяет алгоритм заново.

При получении ссылки на товар, информацию о котором необходимо обновить, модуль загружает страницу интернет-магазина, парсит необходимую информацию: цену без скидки, цену со скидкой, заголовок товара, количество отзывов и его оценку, а также описание товара. Далее создается POST запрос к REST API серверу, и полученная информация передается в контроллер, который обрабатывает данные и сохраняет их в БД.

```
// mvideo_parser.py
answer =
requests.post("http://localhost:8080/api/items/python/update_it
em", json = {
    "title": title,
    "comments": comments,
    "info": info,
    "rating": rating,
    "new_price": new_price,
    "old_price": old_price,
    "link": it.link
}, headers = {"Content-type": "application/json"}).text
items1 = Pykson().from_json(answer, Items)
```

```
print("Обновлена информация для товара с ID = " +  
str(items1.id) + "\n-----")
```

В результате работы модуля информация о ценах и самих товарах своевременно обновляется в БД, что позволяет предоставлять пользователем данные с очень высоким показателем актуальности. Стоить отметить, что новая запись о ценах вносится в базу данных только в том случае, если полученные данные отличаются от последней записи из базы данных. В противном случае система считает, что цена не изменилась, а следовательно, нет необходимости дублировать их еще раз.

4. Тестирование многопользовательского веб-приложения

4.1. Тестирование пользовательского интерфейса веб-приложения

Тестирование пользовательской части проводилось с помощью программы TestComplete. Программа имеет мощный функционал, позволяющий тестировать самые разные приложения, от десктопных до веб-приложений. Отличительной особенностью является то, что при каждом шаге теста сохраняется скриншот тестируемого приложения, что позволяет наглядно увидеть ошибку или неточность в работе алгоритма. Тесты для TestComplete можно записывать через эмулятор нажатий и ввода, либо с помощью кода на одном из языков программирования [10]. С помощью TestComplete были проверены основные алгоритмы регистрации, авторизации, валидации форм, смены пароля пользователя, установки изображения профиля, а также подключения и отключения оповещений в мессенджер Telegram. Выявленные ошибки и баги были исправлены, после чего было проведено повторное тестирование, которое не выявило повторных проблем.

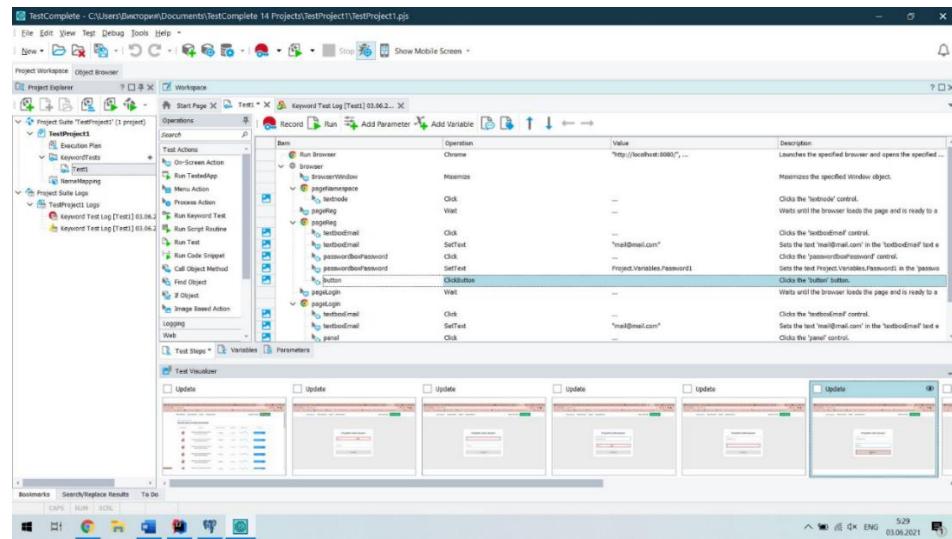


Рисунок 4.1. Окно программы TestComplete

Важной особенностью является возможно просмотреть свойства и структура объектов, с которыми взаимодействует программа. Эта функция

заметно облегчает тестирование приложений, а также позволяет проверить структуру окон вручную.

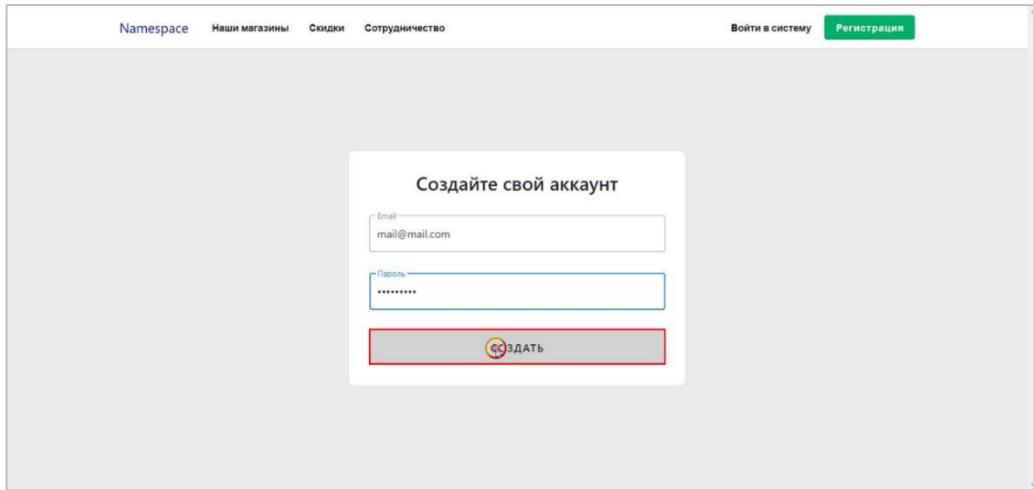


Рисунок 4.2. Изображение с логом теста

```
// TestPasswordForm.tckDTest
function TestPasswordForm()
{
    Browsers.Item(btChrome).Navigate("http://localhost:8080/profile");
    Aliases.browser.BrowserWindow.Maximize();
    Aliases.browser.pageNamespace2.form.passwordboxOld.Click();
    Aliases.browser.pageNamespace2.form.passwordboxOld.SetText(Project.Variables.Password1);
    Aliases.browser.pageNamespace2.form.passwordboxNew.Click();
    Aliases.browser.pageNamespace2.form.passwordboxNew.SetText(Project.Variables.Password2);
    Aliases.browser.pageNamespace2.form.passwordboxConfirm.Click();
    Aliases.browser.pageNamespace2.form.passwordboxConfirm.SetText(Project.Variables.Password2);
    Aliases.browser.pageNamespace2.form.buttonChangePassword.ClickButton();
}
```

Благодаря гибкости TestComplete простые тесты можно записывать всего лишь нажатиями по необходимым элементам. Для более тонкой настройки используется один из предлагаемых языков программирования. В ходе прохождения каждого теста был проверен итоговый лог программы,

позволяющий убедиться в отсутствии каких-либо ошибок или багов. Также были написаны тесты, эмулирующие преднамеренный вызов ошибки, то есть некорректное заполнение форм, авторизация по неправильному паролю или попытка регистрации пользователя с почтовым адресом, который уже присутствует в системе. При прохождении всех этих тестов веб-приложение выводило уведомления об ошибках пользователю, как и было необходимо.

4.2. Тестирование функциональной части веб-приложения

Для тестирования функциональной части веб-приложения для анализа и мониторинга цен интернет-магазинов использовался JUnit. Был проверен функционал создания сущностей и корректности получаемых из них данных, получение и запись данных из БД, уровень привилегий пользователей веб-приложения, соответствие записей из одной таблице записи в другой таблице посредством сопоставления внешнего и первичного ключей. Для создания класса с тестами используется аннотация @Test. Для настройки соединения с базой данных инициализирована библиотека JDBC, с помощью которой можно выполнить произвольный SQL запрос и получить ответ в виде объекта ResultSet.

```
// test/java/Tests.java
@Test
public void createUser() {
    Users users = new Users();
    users.setId(1);
    users.setEmail("admin@admin.ru");
    users.setPassword("Password");
    users.setDisabled(false);
    users.setRole("ROLE_ADMIN");
    users.setImg("Image");

    assertEquals(users.getEmail(), "admin@admin.ru");
}
```

JUnit позволяет покрыть тестами практически весь код, более того с его помощью можно проверять целые функции, компоненты или части кода.

```
// test/java/Tests.java
@Test
public void checkUsersWithAdminRole() {
    PreparedStatement preparedStatement = null;
    int admin = 0;
    try {
        preparedStatement = connection.prepareStatement("SELECT
* FROM USERS");
        ResultSet resultSet = preparedStatement.executeQuery();
        while (resultSet.next()) {
            if (resultSet.getString(5).equals("ROLE_ADMIN")) {
                admin = admin + 1; }
            assertEquals(admin, 1);
        } catch (Exception e) { }
    }
}
```

Все тесты были пройдены успешно, а на основании каждого теста был сформирован лог, который позволяет наглядно убедиться в успешности теста и отсутствии каких-либо проблем или ошибок.

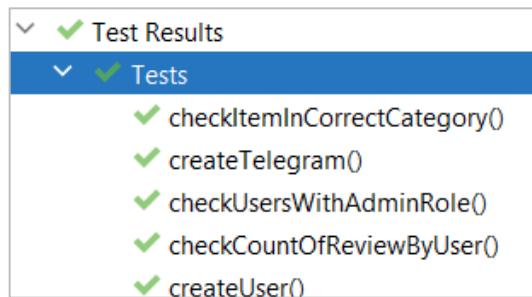


Рисунок 4.3. Лог пройденных тестов

Модули, отвечающие за парсинг страниц интернет-магазинов, были протестированы вручную на большом количестве входных данных, а также с помощью проверки Unit-тестами, после выполнения полной итерации парсинга.

Заключение

В результате выполнения ВКР была изучена и проанализирована предметная область, изучены информационные ресурсы схожей тематики, выявлены недостатки и определены преимущества существующих аналогов. В качестве аналогов были выбраны информационные системы «Яндекс.Маркет» и «Pepper.ru», причиной выбора данных аналогов является популярность среди пользователей. В процессе анализа предметной области было сформировано техническое задание, включающее функциональные требования, требования к интерфейсу и пользовательской безопасности.

Основными параметрами системы являются время отклика на запрос, которое не должно превышать 10-ти секунд, устойчивость системы к некорректным входным данным, количество ошибок при парсинге страниц товаров, не более 0,5% от общего числа запросов, а также частота обновления информации о каждом товаре, интервал между обновлениями не должен превышать 1 час. Была продумана архитектура взаимодействия между различными компонентами веб-приложения, а также спроектирована база данных, состоящая из 14-ти таблиц. Были рассмотрены различные средства разработки и выбраны наиболее подходящие из них. В процессе анализа была составлена ER диаграмма, а также диаграмма последовательности. Был спроектирован пользовательский интерфейс, освоены графические фреймворки BootstrapVue и Vuetyf.

В ходе разработки приложения была успешно применена спецификация JPA, которая позволила упростить работу с запросами к базе данных. Был сконфигурирован полноценный REST API сервер, позволяющий обрабатывать все необходимые запросы через обращение к репозиториям, построенным на спецификации JPA, минуя прямые обращения к базе данных. Реализованы расширенные запросы в каждом из репозиториев, что позволило получать необходимую информацию напрямую из БД без

дополнительной фильтрации данных в самих контроллерах. Был успешно применен фреймворк Hibernate, позволяющий избежать необходимости конфигурации базы данных вручную, что значительно экономит время и позволяет работать преимущественно с объектами. Были успешно настроены файлы конфигурации Spring, а также подключен Spring Security, на основании которого и была построена пользовательская регистрация и авторизация [19]. Разработка системы авторизации с нуля была бы слишком затратна по времени и вложенным силам. Также он был применен для разграничения доступа к различным частям приложения на основании роли пользователя в системе. Была освоена библиотека Vue Resource, позволяющая формировать GET, POST, PUT и DELETE запросы, а также отправлять их на конечные точки API и получать результат от контроллера обратно. Созданные контроллеры позволили настроить адресацию к различным функциям серверной части, все необходимые методы были аннотированы с помощью Spring. Было успешно налажено взаимодействие модулей на Python для парсинга веб-страниц и сервера на Java посредством адаптации классов, а также конфигурации POST и GET запросов к REST API серверу. Разработанные на языке программирования Python модули позволили в полностью автоматическом режиме обновлять данные о ценах в интернет-магазинах, а небольшие интервалы между обновлениями обеспечили высокую актуальность информации.

Пользовательская часть была разделена на отдельные небольшие компоненты, что позволило использовать некоторые из них повторно, избегая дублирования кода в различных компонентах. Были импортированы и использованы дополнительные модули, а также документация к ним. Использование фреймворка. Были обработаны различные типы ошибок со стороны серверной части, а также прописаны механизмы валидации форм с указанием ошибок для пользователя. Была подключена СУБД PostgreSQL, а также настроен файл конфигурации для успешного подключения к ней. В

информационную систему была успешно интегрирована система оповещения пользователей в мессенджере Telegram. В существующий метод был внедрен API мессенджера Telegram. С помощью библиотеки Vue Router было настроено сопоставление компонентов пользовательской части с URL, что обеспечило правильную адресацию в информационной системе.

Разработанное приложение было полностью проверено с помощью различных методов тестирование. Функциональная часть была протестирована с помощью Unit-тестов и ручного тестирования. Пользовательская часть была проверена с помощью современного и мощного инструмента TestComplete, помимо корректных действий были воспроизведены действия с преднамеренными ошибками, для проверки системы уведомления пользователя, предварительной валидации данных, а также отказоустойчивости веб-приложения.

Итоговый объем данных, предоставляемый информационной системой, включает в себя сведения об интернет-магазине, о всех его категориях, о товарах в каждой категории, о самом товаре, включая его изображения, краткие характеристики, оценку в интернет-магазине, а также пользовательские комментарии и оценки, которые могут быть оставлены любым пользователем данного веб-приложения. Благодаря хранению всех данных о цене товара и регулярному автоматическому обновлению этой информации пользователь может наглядно видеть график динамики цены на конкретный товар, что дополнительно повлияет на его выбор перед совершением покупки.

В результате выполнения ВКР были усовершенствованы навыки работы с СУБД PostgreSQL, средой разработки IntelliJ IDEA, менеджером пакетов npm, фреймворком Vue.js и различными библиотеками для Python и Java.

Реализованное приложение имеет большой потенциал для дальнейшего развития и конкуренции с существующими решениями. Объединение

преимуществ аналогов, а также отличительные черты и уникальные функции, которые не реализованы в существующих решениях позволяют предложить хорошую альтернативу пользователям. Автоматизированная информационная система имеет большой потенциал для монетизации, включая продвижение партнерских интернет-магазинов или рекламных материалов.

Список использованных источников

1. Автостопом по Python [Текст]: учебное пособие / Рейтц Кеннет, Шлюссер Таня - СПб.: Питер, 2017. - 336 с.
2. Алгоритмы и структуры данных. Извлечение информации на языке Java [Текст]: учебное пособие / А.Б. Дауни [и др.]. - 2-е изд., перераб. и доп. - СПб.: Питер, 2018. - 240 с.
3. Современные Java-технологии на практике [Текст]: учебное пособие / М.Т. Сергеевич - СПб.: БХВ-Петербург, 2010. - 560 с.
4. Java Persistence API и Hibernate [Текст]: учебное пособие / Кристиан Бауэр, Гэвин Кинг, Гэри Грегори / пер. с англ. Д.А. Зинкевича; под науч. ред. А.Н. Киселева - М.: ДМК Пресс, 2017. - 632 с.
5. Spring 5 для профессионалов [Текст]: учебное пособие / Кларенс Хо, Крис Шефер, Юлиана Козмина, Роб Харроп / пер. с англ. - СПб.: ООО «Диалектика», 2019. - 1120 с.
6. Spring Boot 2: лучшие практики для профессионалов [Текст]: учебное пособие / Гутьеррес Фелипе - СПб.: Питер, 2020. - 464 с.
7. Spring. Все паттерны проектирования [Текст]: учебное пособие / Раджпут Динеш - СПб.: Питер, 2019. - 320 с.
8. Vue.js в действии [Текст]: учебное пособие / Хэнчett Эрик, Листуон Бенджамин - СПб.: Питер, 2020. - 304 с.
9. Быстрый старт — Vuetify.js [Электронный ресурс]. – Режим доступа: <https://v2.vuetifyjs.com/ru/getting-started/quick-start/>. Дата обращения: 10.02.2021.
10. Знакомство с TestComplete 8 [Электронный ресурс]. – Режим доступа: <http://www.interface.ru/home.asp?artId=26912>. Дата обращения: 08.04.2021.

11. Краткое руководство по библиотеке Python Requests [Электронный ресурс]. – Режим доступа: <https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-bibliotekе-python-requests>. Дата обращения: 27.03.2021.
12. Программная навигация [Электронный ресурс]. – Режим доступа: <https://router.vuejs.org/ru/guide/essentials/navigation.html>. Дата обращения: 15.02.2021.
13. Руководство по Spring (полная версия) [Электронный ресурс]. – Режим доступа: <https://proselyte.net/tutorials/spring-tutorial-full-version/>. Дата обращения: 25.01.2021.
14. Getting Started — BootstrapVue [Электронный ресурс]. – Режим доступа: <https://bootstrap-vue.org/docs/>. Дата обращения: 15.02.2021.
15. Lingallery [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/lingallery>. Дата обращения: 04.03.2021.
16. Maven Guide [Электронный ресурс]. – Режим доступа: <http://maven.apache.org/guides/getting-started/index.html>. Дата обращения: 02.02.2021.
17. Pykson: A JSON Serializer/Deserializer for Python [Электронный ресурс]. – Режим доступа: <https://pypi.org/project/pykson/>. Дата обращения: 03.04.2021.
18. Spring MVC — основные принципы [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/336816/>. Дата обращения: 07.02.2021.
19. Spring MVC и Spring Security. Пример настройки страницы логина, настройка ролей [Электронный ресурс]. – Режим доступа: <https://javastudy.ru/spring-mvc/security>. Дата обращения: 14.02.2021.
20. Spring Security [Электронный ресурс]. – Режим доступа: <https://docs.spring.io/spring-security/site/docs/3.0.x/reference/springsecurity.html>. Дата обращения: 10.02.2021.
21. Vue Sidebar Menu [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/vue-sidebar-menu>. Дата обращения: 19.02.2021.

22. Vue Trend Chart [Электронный ресурс]. – Режим доступа:
<https://github.com/dmtrbrl/vue-trend-chart>. Дата обращения: 17.03.2021.

ER-диаграмма базы данных

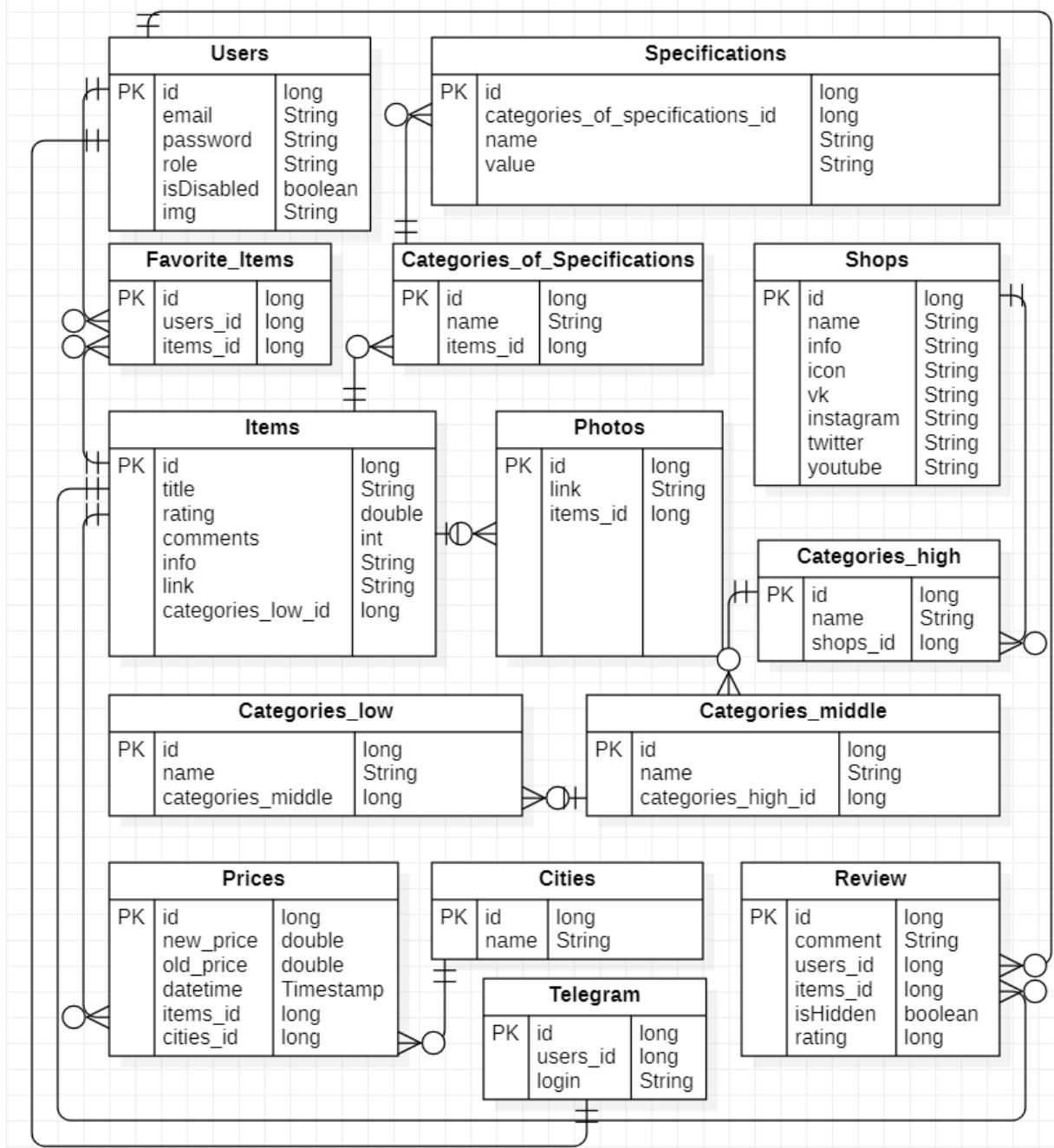


Рисунок А.1. ER диаграмма базы данных

Зависимости файла конфигурации pom.xml

```
<!-- pom.xml  
Зависимости для работы проекта -->  
<dependencies>  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-thymeleaf</artifactId>  
    </dependency>  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-web</artifactId>  
    </dependency>  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-test</artifactId>  
        <scope>test</scope>  
        <exclusions>  
            <exclusion>  
                <groupId>org.junit.vintage</groupId>  
                <artifactId>junit-vintage-engine</artifactId>  
            </exclusion>  
        </exclusions>  
    </dependency>  
    <dependency>  
        <groupId>org.springframework.security</groupId>  
        <artifactId>spring-security-test</artifactId>  
        <scope>test</scope>  
    </dependency>  
    <dependency>  
        <groupId>junit</groupId>  
        <artifactId>junit</artifactId>  
        <version>4.12</version>  
    </dependency>  
    <dependency>  
        <groupId>org.postgresql</groupId>  
        <artifactId>postgresql</artifactId>  
        <version>42.2.8</version>  
    </dependency>  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-data-jpa</artifactId>  
    </dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
</dependencies>
```